Ketterät ohjelmistomenetelmät startup-yrityksen innovaatioprosessissa

Tapio Heiskanen

Helsinki 14.2.2013

Pro gradu -tutkielma

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta Fakultet – Faculty	Laitos Institution Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä Författare Author

Tapio Heiskanen

Työn nimi Arbetets titel Title

Ketterät ohjelmistomenetelmät startup-yrityksen innovaatioprosessissa

Oppiaine Läroämne Subject
Tietojenkäsittelytiede

Työn laji Arbetets art Level	Aika Datum Month and year	Sivumäärä Sidoantal Number of pages
	02-2013	80 sivua

Tiivistelmä Referat Abstract

Ohjelmistoala on yksi innovatiivisimmista teollisuuden aloista. Innovaatiolla tässä tutkielmassa tarkoitetaan uusien tuotteiden, palveluiden, tuotantotavan tai -menetelmän esittelemistä. Innovaation ei siis välttämättä tarvitse olla luova, vaan tehostettu tapa tehdä jotain luovaa. Innovatiivisuudella tarkoitetaan kykyä kääntää luovuus todelliseksi tuotteeksi, palveluksi tai käytännöksi.

Ohjelmistotuotteita valmistavan yrityksen työntekijät kohtaavat asiakkaitaan eri tilanteissa. Kukin työntekijä on kosketuksissa joko asiakkaiden, käyttäjien tai potentiaalisten asiakkaiden kanssa, jolloin asiakkailta saadaan palautetta koko ohjelmistotuotteen elinkaaren aikana. Asiakas- ja käyttäjäinnovaatioiden merkitys ohjelmistotuotteiden valmistamisessa on korostunut entisestään, ja näin ollen innovaatiot ovat otettava huomioon itse ohjelmistotuotteen lisäksi myös sen kehitysprosessissa.

Tutkielmassa tutustutaan olemassa oleviin innovaatiomenetelmiin, jotka soveltuvat ohjelmistoalalle. Tutkielmassa perehdytään yleisten olemassa olevien ohjelmistoprosessien tarkasteluun innovaatiomenetelmiä hyödyntävän startup-yrityksen, AdPearl Oy:n, vaatimusten näkökulmasta. Nykyisten menetelmien lisäksi AdPearl Oy:lle esitellään uusi Lean Startup -menetelmään pohjautuva, yrityksen tarpeisiin räätälöity, innovaatiomenetelmiä hyödyntävä ohjelmistokehitysprosessi.

Avainsanat – Nyckelord Keywords
Lean Startup, Startup, Ohjelmistokehitysprosessi, Innovaatiot, Extreme Programming, Scrum, Kanban, Feature Driven Development

Säilytyspaikka Förvaringställe Where deposited

Muita tietoja Övriga uppgifter Additional information

Sisältö

Lyhennelista

Lyhenne	**Merkitys**	**Suomeksi**
B2B	Business To Business	Yritykseltä yritykselle
B2C	Business To Customer	Yritykseltä kuluttajalle
CTR	Click Trough Rate	Klikkausprosentti
FDD	Feature Driven Development	Ominaisuusvetoinen kehitys
KISS	Keep It Short and Simple	Kaikesta pitäisi tehdä niin yksinkertaista kuin mahdollista
MVP	Minium viable product	Minimaalinen toimivan tuote
TDD	Test Driven Development	Testivetoinen kehitys
XP	Extreme Programming	Extreme Programming

1 Johdanto

Innovaatioiden ja innovatiivisuuden arvostus länsimaissa on kasvanut globaalin kilpailun ja kehittyvien maiden alhaisempien valmistuskustannuksen sekä nopean teknologiakehityksen myötä. Yrityksen kyky tuottaa uusia innovaatioita on noussut keskeiseksi tekijäksi tulevaisuuden menestyksen turvaamiseksi. Tämä on johtanut siihen, että myös innovaatioprosesseista on tullut yksi yritysten pääprosesseista yksittäisen tuotekehitystoiminnan sijaan [ApT06].

Innovaatioille on myös asemansa ohjelmistoteollisuudessa: Ohjelmistoinnovaatiot ovat yksi keino edistää korkeamman arvon tuotteiden ja palveluiden toteuttamista. Innovaatioita voi tapahtua ohjelmistotuotteen lisäksi myös sen kehitys- ja jakeluprosessissa.

Innovaatiolla tässä tutkielmassa tarkoitetaan uusien tuotteiden, palveluiden, tuotantotavan tai -menetelmän esittelemistä. Innovaation ei siis välttämättä tarvitse olla luova, vaan tehostettu tapa tehdä jotain luovaa. Innovatiivisuudella tarkoitetaan kykyä kääntää luovuus todelliseksi tuotteeksi, palveluksi tai käytännöksi.

Tässä tutkielmassa olemassa olevista innovaatiomenetelmistä esitellään suunnitteluajattelu, palvelumuotoilu, avoin innovointi, tutustutaan käyttäjäinnovaatioihin sekä esitellään Millerin innovaatioprosessi.

Startup-yritys on yleensä ensimmäistä tuotettaan kehittävä nuori, innovatiivinen, nopeaa kasvua tavoitteleva ja skaalautuvalla liiketoimintamallilla operoiva yritys. Startup-yritykselle on myös tyypillistä, että sillä ei ole vielä kannattavaa liiketoimintaa. Lean Startup -menetelmä on Erin Riesin esittelemä, nopeaan markkinaoletusten vahvistamiseen tähtäävä menetelmä startup-yrityksille. Menetelmän ydin on toteuttaa mahdollisimman nopeasti oikeille asiakkaille suppein mahdollinen toimiva tuote, jonka avulla tuotteen toimivuudesta voidaan oppia projektin aikaisessa vaiheessa.

Ensimmäistä tuotettaan kehittäville Startup-yrityksille epävarmuus on yhdistävä tekijä. Perinteiset vesiputousmalliin pohjautuvat menetelmät eivät ole parhaimmillaan vastaamaan äkillisiin muutoksiin. Vuosituhannen vaihteessa perinteisten ohjelmistokehitysprosessien vastapainoksi esiteltiin uudenlaisia, ketteryyteen ja pikaisiin muutoksiin kykeneviä ketteriä menetelmiä (Agile methods). Ketterille menetelmille ominaisia piirteitä suhteessa perinteisiin ohjelmistokehitysmalleihin ovat

kevyempi dokumentaatio, iteratiivisuus, inkrementiaalisuus, ihmiskeskeinen lähestymistapa ja kyky nopeisiin muutoksiin.

Ketteriä menetelmiä on useita. Tässä tutkielmassa tutustutaan tarkemmin ketteristä menetelmistä suosituimpiin, Extreme Programming (XP), Scrum, Kanban ja Feature Driven Development -menetelmiin. Esitetyt ketterät menetelmät ovat pääosin Ketterän Manifestin (Agile Manifest) määritelmän periaatteiden mukaisia, tarjoten kukin omat ominaispiirteensä.

Tutkielmassa perehdytään kirjallisuuden avulla siihen, kuinka ketterät menetelmät sopivat startup-yrityksen innovaatioprosesseihin. Menetelmien soveltuvuutta arvioidaan AdPearl Oy:n asettamien liiketoimintamallin sekä innovaatiovaatimusten näkökulmasta. Tarkoituksena on kartoittaa kirjallisuuden avulla esiteltävien ketterien menetelmien ominaispiirteet ja peilata menetelmän soveltuvuutta AdPearl Oy:n ohjelmistoprosessille asettamiin vaatimuksiin. Tutkielman tutkimuskysymyksiä ovat:

1. Kuinka ketterät menetelmät sopivat yhteen startup-yritysten innovaatioprosssien kanssa?
2. Mikä ketterä menetelmä sopii ominaisuuksiltaan parhaiten?
3. Millainen ketterä menetelmä tukee innovaatioprosesseja parhaiten?

Tutkielma koostuu johdannosta, kuudesta sisältöluvusta ja yhteenvedosta. Toisessa luvussa tarkastellaan olemassa olevia innovaatiomenetelmiä, Lean Startup -menetelmää, sekä yleisesti ketteriä menetelmiä. Kolmannessa luvussa kuvataan AdPearl Oy:n liiketoimintamalli sekä vaatimukset ohjelmistokehitysprosessille. Neljännessä luvussa esitellään XP, Scrum, Kanban ja FDD-menetelmät tarkastelleen niiden soveltuvuutta AdPearl Oy:lle. Viidennessä luvussa esitellään AdPearl Oy:lle räätälöity ohjelmistokehitysprosessi, joka pohjautuu Lean Startup -menetelmään. Kuudennessa luvussa analysoidaan tutkielman tuloksia. Viimeisessä luvussa vedetään tutkielman tulokset yhteen.

2 Innovointi ohjelmisto-startup -yrityksissä

Ohjelmistojen käyttäjäryhmä on muuttunut paljon ohjelmistoteollisuuden alkuajoista. Nykyään – toisin kuin esimerkiksi 60-luvulla – ohjelmistojen käyttäjäkunta on monipuolinen: ohjelmistoja käyttävät lähes kaikki, tiedostaen tai tietämättään, ilman erityistä osaamista tietojenkäsittelyn alalta. Useasti perinteinen rutiininomainen ohjelmistotyö ulkoistetaan sinne, missä ohjelmistot on halvinta toteuttaa. Länsimaisten yritysten on vaikea kilpailla kehittyvien maiden kanssa kilpailukyvyssä. Globalisaatio ja teollistuminen asettavat länsimaisille ohjelmistoyrityksille haasteen: rutiininomaisen ohjelmistojen tuottamisen sijaan on kehitettävä korkeamman arvon palveluita tai tuotteita. Ohjelmistoinnovaatiot ovat yksi keino edistää korkeamman arvon tuotteiden ja palveluiden tuottamista. Ohjelmistotalojen on oltava innovatiivisia erottuakseen ohjelmistomarkkinoilla [SDW04].

Innovaatiovetoisen ohjelmistokehitys pitää sisällään erityisominaisuuksia. Tarkkaa etukäteistä suunnittelua ja raamittamista vaativat prosessimallit eivät sovi ohjelmisto-startup -yrityksen tarpeisiin [Rie11]. Ohjelmistoprosessin on oltava ketterä ja sopeutuva pikaisille muutoksille. Tyypilliset innovaatiovetoisen ohjelmistokehityksen erityisvaatimukset on kuvattu aliluvussa 2.2.

Startup-yritykset ovat luonteeltaan innovatiivisia [Rie11]. Tällaiset yritykset operoivat epävarmoissa olosuhteissa, jolloin niiden harjoittama ohjelmistokehitys vaatii tietyt ominaispiirteensä. Aliluvussa 2.3 käsitellään startup-yrityksiä ja tutustutaan Lean Startup -menetelmään, joka mahdollistaa markkinointioletusten nopean testaamisen prototyypeillä.

2.1 Innovaatiot ja innovointi

Innovatiivisuuden merkityksen ymmärtäminen on tärkeää myös yksittäisen ohjelmistokehittäjän kannalta. Tietotaitoiset ohjelmistokehittäjät kaipaavat haasteita itseään toistavan ja rutiininomaisen ohjelmoinnin sijaan. Pitääkseen itsensä kilpailukykyisinä työntekijöinä, on myös ohjelmistokehittäjien oltava innovatiivisia [SDW04].

Everett Rogers määrittelee innovaation sellaiseksi esineeksi, käytännöksi tai ideaksi,

jota käyttäjät pitävät uutena [Rog62]. Joseph Schumpeter määrittelee innovaation seuraavasti [Sch39]:

1. Innovaatio on sellaisen uuden tuotteen tai palvelun esittely, joka ei ole kuluttajille entuudestaan tuttu.
2. Innovaatio on uuden tuotantotavan tai -menetelmän esittely, jonka ei tarvitse millään muotoa olla tieteellisesti uusi, ja joka voi olla myös uusi tapa kaupallistaa hyödyke.
3. Innovaatio on markkinan avautuminen - sellaisen, jossa tuotetta ei aikaisemmin ole ollut kaupan, olivatpa nämä markkinat olleet olemassa jo aikaisemmin tai eivät.
4. Innovaatio on uuden raaka-aineen ja puolivalmisteen toimituslähteen haltuunotto, jälleen riippumatta siitä, oliko toimituslähde ollut olemassa jo aikaisemmin vai luotiinko se ensimmäistä kertaa.
5. Innovaatio on uuden teollisen markkinarakenteen toteuttaminen, kuten monopoliaseman luominen tai purkaminen.

Innovaation ei siis välttämättä tarvitse olla luova, vaan tehostettu tapa tehdä jotain luovaa. Åbo Akademin professorin Alf Rehnin mukaan *luovuus* on kyky nähdä jotain mitä muut eivät kykene näkemään. Luovuus on kyky murtaa olemassa olevia malleja - tuomaan jotain uutta. *Innovatiivisuus on* taas kyky kääntää luovuus todelliseksi tuotteeksi, palveluksi tai käytännöksi [Reh11].

Innovaatiotutkimus yleisellä tasolla on kehittynyttä ja ohjelmistoalalle sovellettavaa. Ohjelmistoalalle soveltuvia menetelmiä innovaatioiden aikaansaamiseksi ovat esimerkiksi suunnitteluajattelu, palvelumuotoilu, käyttäjäinnovaatiot, avoin innovointi ja Lean Startup -menetelmä.

2.1.1 Suunnitteluajattelu

Suunnitteluajattelun (Design thinking) idea on tuotetta, palvelua tai prosessia kehitettäessä ajatella kuin suunnittelija, joka ottaa huomioon asiakkaiden tarpeet, käyttökontekstin ja kulttuurin. Suunnitteluajattelu pyrkii ihmisen ja sen toimintaympäristön läpikotaiseen tuntemiseen [Bro08].

Suunnitteluajattelun tavoitteena on suunnitella tuote tai palvelu lähtökohtaisesti vastaamaan kuluttajien tarpeisiin. Pyrkimyksenä on tietää, mistä kuluttajat pitävät ja mistä he eivät pidä, mitä he haluavat ja mitä he tarvitsevat elämässään. Asiakkaan

tuntemisen kannalta merkittäviä tekijöitä ovat esimerkiksi sellaiset seikat, kuin miten tuote kannattaa tehdä, myydä, paketoida, markkinoida ja tukea, jotta kehitettävä tuote tai palvelu vastaa asiakkaan tarpeisiin kokonaisvaltaisesti [Bro08].

Suunnitteluajattelun lähtökohta on olla ihmiskeskeinen tapa rakentaa tuotetta tai palvelua. Keskeisiä kysymyksiä ovat

- mitä ihmiset tarvitsevat tai voivat tarvita,
- mikä tekee elämästä helpompaa ja nautittavampaa,
- missä on ongelmia,
- missä mahdollisuuksia,
- mikä on muuttunut tai muuttumassa ja
- mikä tekee teknologiasta hyödyllistä ja käytettävää.

Kysymys ei niinkään ole tuotteen ulkomuodosta, vaan tuotteen tai palvelun käyttökontekstin ja kulttuurin ymmärtämisestä.

Suunnitteluajattelu keskittyy passiivisen tuottaja-kuluttaja -suhteen sijaan hyödyntämään kuluttajien osallistumista tuotekehityksen aikana. Kaikkien kuluttajien kokemusten kartoittaminen on olennaista oikeanlaisen tuotteen saavuttamiseksi. Parhaiden olemassa olevien mahdollisuuksien valitsemisen sijaan, suunnitteluajattelu kannustaa etsimään uusia vaihtoehtoja, uusia ratkaisuita ja uusia ideoita joita ei vielä ole olemassa. Kehitysprosessi on iteratiivinen ja se koostuu kolmesta vaihteesta [Bro08]: *inspiroitumisesta* (inspiration), *ideoimisesta* (ideation) ja *toteuttamisesta* (implementation). Inspiroitumisen kannalta on olennaista odottaa oman tuotteensa tai palvelun menestyvän. Tässä vaiheessa pyritään rakentaa resursseja siihen, että tuote tai palvelu saadaan toteutettua parhaalla mahdollisella tavalla. Keinoja resurssien luomiseen on tarkkailla tulevien käyttäjien toimintatapoja, ajatuksia ja tarpeita, määrittää liiketoimintarajoitteita, kiinnittää tarkkaa huomiota ”äärimmäisiin” käyttäjiin, kuten lapsiin ja vanhuksiin, sekä miettiä millainen teknologia auttaa käyttäjiä parhaiten. Ideointivaiheessa pidetään aivoriihiä ideoiden synnyttämiseksi, toteutetaan luonnoksia tulevasta tuotteesta tai palvelusta, asetutaan tulevien käyttäjien asemaan sekä luodaan alustavia prototyyppejä. Toteuttamisvaiheessa luodaan prototyyppejä ja testataan niitä niin sisäisesti kuin kuluttajienkin kesken.

Jatkuva prototyyppien tuottaminen on toimiva keino innovatiivisen prosessien

vauhdittamiseksi. Prototyyppien avulla ideoiden vahvuudet ja heikkoudet paljastuvat. Mitä nopeammalla syklillä uusia prototyyppejä tuotetaan, sitä nopeammin ideat jalostuvat. Sen sijaan, että ajatellaan mitä ollaan rakentamassa, tulee rakentaa ajattelun ehdoilla [Bro08].

2.1.2 Palvelumuotoilu

Palveluilla on merkittävä rooli nyky-yhteiskunnassa. Palveluiden kirjo on runsas ja omien palveluiden erottuminen kilpailijoiden palveluista vaatii uusia toimintatapoja suhteessa kilpailijoihin. *Palvelumuotoilu* (Service design) hyödyntää muotoilun menetelmiä palveluiden kaupallisessa kehittämisessä [SDN12a]. Palvelumuotoilu keskittyy palveluiden suunnittelemiseen asiakkaiden perspektiivistä.

Tässä tutkielmassa tuote on sellainen ohjelmistojakelu, jonka pystyisi pakkaamaan pakettiin ja myymään kaupan hyllyltä. Esimerkki tällaisesta tuotteesta on käyttöjärjestelmä tai kuvankäsittelyohjelma. Palvelu on puolestaan jatkuvasti elävä, interaktiivisempi ohjelmistotuote, esimerkiksi kaikki sosiaalisen median palvelut, kuten *Twitter* ja *Facebook*.

Palvelumuotoilu pyrkii varmistamaan, että asiakkaat kokevat palvelun hyödyllisenä ja käytettävänä, ja palveluntarjoaja kokee palvelun tehokkaana, vaikuttavana ja erikoislaatuisena. Palvelumuotoilijat suunnittelevat palveluita, jollaisia ei välttämättä ole vielä olemassa: he tarkkailevat ja tulkitsevat palveluiden tarpeita ja asiakkaiden käyttäytymismalleja muuttaen ne uusiksi palveluiksi. Olemassa olevien palveluiden uudistaminen vaikeaa, kuten myös täysin uusien innovatiivisten palveluiden luominen [SDN12a].

Palvelumuotoilu on järjestelmällinen ja iteratiivinen prosessi [SDN12b]. Palvelumuotoilun lähtökohta on ajatella palvelua tuotteena ja keskittyä asiakkaan kokemaan hyötyyn. Ideana on kehittää konkreettinen palveluprosessi, joka keskittyy arvon tuottamiseen asiakkaalle parhaalla mahdollisella tavalla. Prosessin kehittämisessä on olennaista, että sidosryhmien edustajat ovat mukana sitä kehittämässä.

Palvelu nähdään kokonaisuutena: palvelukokemus voi alkaa jo kauan ennen kuin asiakas on yhteydessä palvelun tarjoajaan – ja palvelukokemus voi jatkua vielä pitkään yhteydenpidon päättymisen jälkeen. Palvelu on suunniteltava siten, että se on asiakkaalle kokemus. Palvelun laatu on oltava selkeästi mitattavissa. Palvelukokemuksen mittaamisen tulee olla selkeitä mittareita. Palveluilla pitää olla

myös kykyä muuttua ajan myötä [SDN12b].

Palvelu on holistinen kokonaisuus, osiensa summa. Kaikki palveluun vaikuttavat osapuolet on tunnistettava, ja osapuolten välisten suhteiden kuvaaminen on tärkeää. S*idosryhmäkartoilla* (stakeholder maps) kuvataan kaikkien osapuolten suhteet. Sidosryhmäkartan jokaista suhdetta tarkastellaan yksitellen palvelun tarjoajan perspektiivistä: mitä arvoa palveluntarjoaja tuottaa osapuolelle, ja mitä arvoa osapuoli tuottaa palveluntarjoajalle [SDN12b].

2.1.3 Avoin innovointi

Avoin innovointi (open innovation) on menetelmä, jossa yritykset käyttävät omien sisäisten ideoiden lisäksi myös ulkoisia ideoita, ja jakavat omia ideoitaan ulkopuolisille. Avoin innovointi on sisäisen vertikaalisen integraation vastakohta, jossa yritys usein kontrolloi merkittävää osaa tuotantoketjunsa osista. Vertikaaliselle integraatiolle on tyypillistä, että kaikki sisäinen tutkimus- ja kehitystyö johtaa sisäisesti kehitettyihin tuotteisiin jotka yritys itse jakelee [Che05]. Avointa innovointia harrastava yritys voi esimerkiksi jakaa omaan strategiaansa sopimattomia ideoita ja teknologioita ulkopuolisille tahoille jatkokehitystä tai lisensointia varten, hyötyen siitä myös itse.

Avoimesti innovoiva yritys on myös avoin ulkopuolisille ideoille [Che05]. Yritys tiedostaa, että kaikki älykkäät ihmiset eivät suinkaan ole töissä ainoastaan heidän yrityksessään. Ohjelmistotalot voivat esimerkiksi panostaa avoimen lähdekoodin projekteihin ja hyödyntää sen aikaansaannoksia omissa tuotteissaan. Vaikutteita voidaan ottaa myös eri yliopistoissa tehdyistä tutkimuksista.

Esimerkki erinomaisesti menestyneestä avoimesta innovoinnista on yhdysvaltalaisen IT-alan yhtiö *Applen* konseptoima *App Store*, joka hoitaa Applen omille laitteille tehtävän ohjelmistojakelun. Applen laitteet yhdessä App Storen kanssa luovat ekosysteemin, jossa Applen omat tuotteet ovat keskiössä. Laitteiden menekki on laaja, kun ulkopuoliset yritykset tuottavat omia innovatiivisia ohjelmistojaan Applen laitteille [BoL09]. Tällaisessa tapauksessa avoimen innovoinnin periaate toteutuu: ulkopuolinen yritys luovuttaa omia ideoitaan toisen yrityksen käyttöön, hyötyen niistä samalla itse.

Yhteisöissä avointa innovaatiota on harrastettu jo pidempään [BoL09]. Tunnettu esimerkki yhteisön yhdessä kehittämästä laajasta ohjelmistosta on Linux-käyttöjärjestelmä, jonka juuret ulottuvat 1990-luvun alkupuolelle. Yhteisöjen piirre, jossa yhteisön jäsenet ovat valmiita jakamaan tietotaitoaan jopa ilmaiseksi, ei

toistaiseksi ole ollut helppoa saavuttaa yritysmaailmassa: ulkoisten innovaattorien motivoiminen on iso haaste.

Innovaatioiden ”ulkoistaminen” vaatii pohdintaa kolmelta eri alueelta [BoL09]:

1. minkä tyyppisiä innovaatioita ulkoistetaan,
2. mikä motivoi ulkoisia innovaattoreja ja
3. mikä on yrityksen liiketoimintamalli.

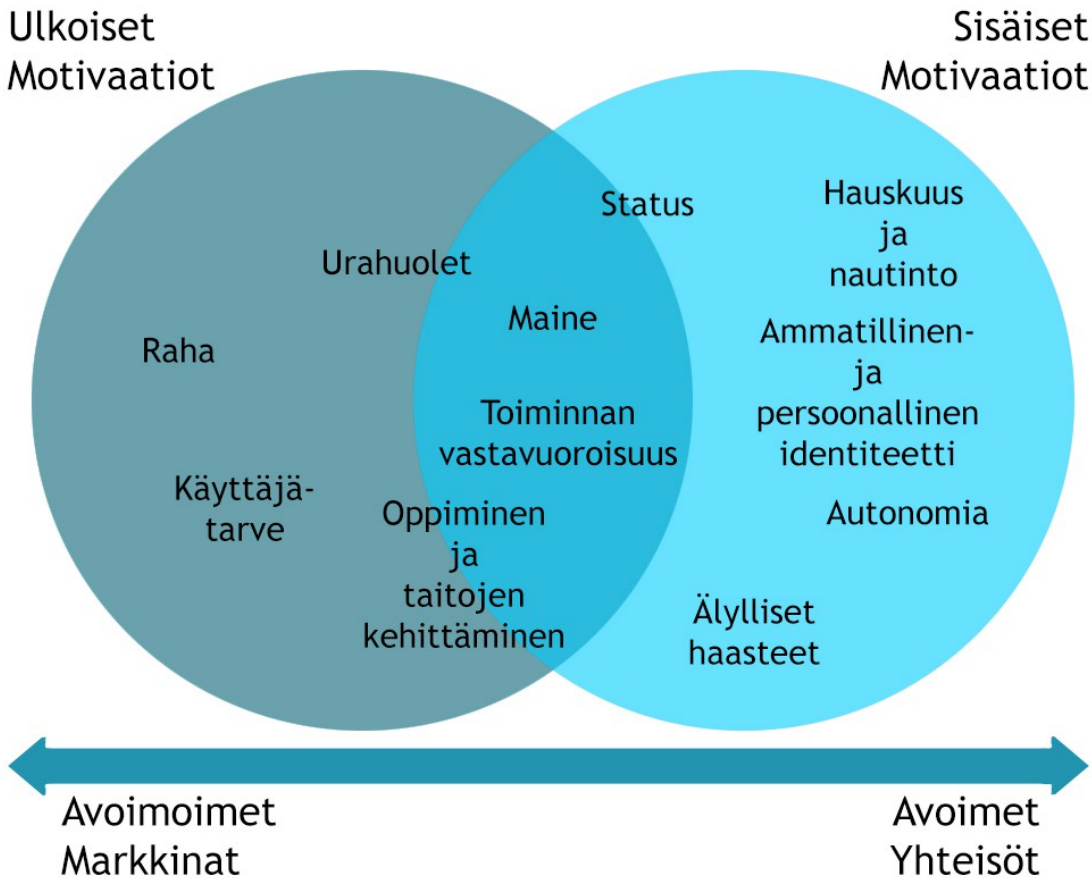
Innovaatioiden ulkoistamisesta voi olla yritykselle merkittäviä hyötyjä [BoL09]. Kun asiakkaan tarpeet vaihtelevat tai niitä ei vielä täysin tiedetä, voi innovaatioiden ulkoistaminen olla järkevää. Innovaatioiden ulkoistaminen on hyödyllistä etenkin silloin, kun yritys voi täysin ulkoistaa jonkin tietyn osa-alueen omasta palvelustaan tai tuotteesta. Sitä, ulkoistaako innovaatioiden tuottamisen yhteisöille vai yrityksille, on mietittävä tarkkaan.

Jos innovaatio-ongelma vaatii pitkäjänteistä, aiempaan työhön perustuvaa tietoa, yhteisöille ulkoistaminen on kannattavaa [BoL09]. Yhteisöt ovat luonnostaan orientoituneita toteuttamaan sellaisia ratkaisuita, jotka vaativat integrointikykyä ja teknologioiden hyväksikäyttöä. Yhteisöille on luontevaa jakaa ja levittää omaa tietoaan, sekä tehdä yhteisötyötä.

Jos innovaatio-ongelma ratkaistaan parhaiten laajojen asiakasryhmien tai teknisillä menetelmillä, yritykselle ulkoistaminen on kannattavaa. Yhteisöillä on tapana tehdä oletuksia siitä, mikä tehtävä on saatettu loppuun asti ja mikä ei. Yritysmaailmalle tämä on vieraampaa: niillä on tapana edistää erilaisuutta, rohkaista kokeilemaan ja kannustaa omaperäisyyteen. Yritykset haluavat lähtökohtaisesti edistää toteuttamiaan teknologioita ja pitää kiinni patentoimistaan tuotteista erottuakseen kilpailijoista [BoL09].

Johtajien on myös mietittävä, mikä motivoi ulkoisia innovaattoreita. Ulkoisten innovaattoreiden motivaatiot ovat mitä erilaisimpia [Pin09]. Motivaatiot voidaan jakaa karkeasti kahtia: ulkoisiin ja sisäisiin motivaatioihin [Pin09]. Ulkoisia motiiveja ovat mm. raha, käyttäjien tarve, urahuolet, oppiminen ja taitojen kehittäminen. Sisäisiä motiiveja ovat hauskuus ja nautinto, ammatillinen ja persoonallinen identiteetti sekä älylliset haasteet. Maine ja toiminnan vastavuoroisuus voidaan nähdä kuuluvan molempiin. Kuvasta 1 on nähtävissä kuinka ulkoiset motivaatiot liittyvät useammin yritysmaailmalle tyypillisiin motivaatioihin ja kuinka sisäiset motivaatiot ovat

yleisempiä avoimissa yhteisöissä. Motivaatioiden monimuotoisuuden takia, yrityksen on harkittava tarkkaan valittaessa yhteisöjen ja yritysten välillä [BoL09]. Projektipäälliköiden on syytä kehittää sellaiset toimintatavat jotka käyttävät hyväksi projektiin haluttavien jäsenten motivointikeinoja. Valinta vaikuttaa täten innovointiin osallistuvien henkilöiden työpanokseen ja prosessille omistautumiseen.



Kuva 1: Sisäisten ja ulkoisten motiivien jaottelu [BoL09].

Kun yritys avaa tuotteensa ulkoisille innovaattoreille, tuotteesta tulee *alusta* (platform). Kun tuote on alusta, silloin yritys mahdollistaa ulkoisten innovaattorien tuottaa sisältöä heidän tarjoamalleen alustalle, jolloin sekä yritys että ulkoinen innovaattori hyötyvät. Tuotteen muuttaminen liikevaihtoa tuottavaksi alustaksi vaatii selkeää liiketoimintamallia. Kysymys ”kuka myy kenelle?” auttaa ymmärtämään, kumpi kahdesta vaihtoehdosta, yritysmaailmasta ja yhteisöistä, kannattaa valita ulkoisten innovaatioiden lähteiksi liiketoiminnan näkökulmasta. Kysymys määrittää, kuka päättää

teknologisen kehityksen suunnasta, tulovirrasta, suhteesta loppukäyttäjiin sekä siitä, kuinka paljon ulkopuolisilla innovaattoreilla itsemääräämisvaltaa. Kuka-myy-kenelle -asetelman kannalta alustan liiketoimintamalli voidaan jakaa kolmeen luokkaan: *yhdentäjä*-, *tuote*- ja *kaksipuoliseen* alustaan [BoL09].

Yhdentäjäalustassa yritys yhdistää ulkoiset innovaatiot ja myy lopullisen tuotteen asiakkaille. Esimerkiksi Applen *iPhone* toimii yhdentäjäalustana yhdistämällä asiakkaat ja ohjelmistokehittäjät toisiinsa App Storen kautta. Applen on mahdollista suoraan tarkkailla ja kontrolloida transaktioita ottaen 30% provision kaikesta App Storen kautta tapahtuvasta myynnistä.

Tuotealusta mallissa ulkoiset innovaattorit rakentavat alustan päälle ja myyvät aikaansaadut tuotteet asiakkaille. Yrityksillä on tässä mallissa vähemmän määräysvaltaa. Alustan omistaja voi tehdä suoraan sopimuksen ulkoisten innovaattoreiden kanssa määräten epäsuorasti ulkoisia innovaattoreita alustan ydinteknologian kautta. Esimerkki tuotealustasta on mikroprosessorivalmistaja *Intelin* ”Intel inside”-malli jossa Intel tarjoaa ydinteknologian ja määrää sen käytön, ja lisenssinhaltijat innovoivat alustan päälle ja myyvät omia tuotteitaan asiakkaille.

Kaksipuolisessa alustassa ulkoiset innovaattorit ja asiakkaat ovat vapaita harjoittamaan liiketoimintaa suoraan toistensa kanssa, kunhan he liittyvät tavalla tai toisella myös alustan omistajan kanssa. Tällaisessa tapauksessa alusta helpottaa transaktioita ja yhteydenpitoa kahden osapuolen välillä, ilman että osapuolten tarvitsee olla suorassa yhteydessä alustan tarjoajan kanssa suunnitellessaan, kehittäessään ja valmistaessaan uutta tuotetta. Esimerkki kaksipuolisesta alustasta on *Facebook*. Facebook on sosiaalisen median alusta, jonka kautta käyttäjät voivat olla vuorovaikutuksessa kolmannen osapuolten sovelluksen kanssa. Sovellukset voivat sijaita Facebookista erillisessä ympäristössä. Kaikki vuorovaikutus tapahtuu erillisestä ympäristöstä huolimatta Facebookin alustan kautta. Myös pelikonsolit noudattavat kaksipuolisen alustan mallia: pelaajat voivat pelata pelivalmistajien itsensä hinnoittelemia pelejä peleille soveltuvalla alustalla (pelikonsolilla). Kaksipuolisessa alustassa ulkoiset innovaattorit voivat vapaasti hinnoitella tuotteensa tai palvelunsa.

2.1.4 Käyttäjäinnovaatiot

Käyttäjät ovat monien innovaatioiden lähteenä [MAL08]. Käyttäjien innovointi on usein tiedostamatonta. Käyttäjät myös luovat uusia innovaatioita käyttämällä tuotteitta ennustamattomalla tavalla.

Monet innovaatiot perustuvat sen ymmärtämiseen, mikä tuo asiakkaalle arvoa. Useat innovaatiot syntyvätkin käyttäjien tai harrastajoiden toimesta. *Harrastelijatiedolla* (Hobbyist knowing) tarkoitetaan sellaista tietoa tuotteen tai palvelun käyttökohteesta, joka on syntynyt pitkäaikaisen harrastustoiminnan kautta [Kot07]. Harrastelijatieto voi parhaimmillaan auttaa ymmärtämään käyttäjien tarpeita ja vaatimuksia. Harrastelijatietoa omaavat tuotekehittäjät ovat samalla yrityksen työntekijöitä ja kehitettävän tuotealueen harrastajia. Tällaiset työntekijät ovat kuin *lead-käyttäjiä*, jotka työskentelevät yrityksen sisällä [Kot07]. Yrityksen on pyrittävä tunnistamaan mitä epäsuoraa tietoa tuotteen käyttäjistä löytyy yrityksen sisältä ja kuinka tällainen tieto toimii yrityksen sisällä. Yksityiselämän mielenkiinnon kohteet vaikuttavat työelämän suorituksiin: mielenkiinnon kohteet seuraavat myös työelämään [Kot07].

Suomalainen paikannuslaitteita ja sukellustietokoneita valmistava yhtiö *Suunto* hyödyntään tuotekehityksessään työntekijöiden urheilutaustaa [Kot07]. Urheilutaustaa omaavat työntekijät tuovat oman kokemuksensa tuotekehitykseen, sekä toimivat yhtenä elementtinä yrityksen brändin kehittämisessä. Työn ulkopuolisella, aamiaisella, lounaalla tai kahvitauolla tapahtuvalla, *tarinankerronnalla* on tärkeä rooli. Tarinankerronta on tehokas tapa ymmärtää mitä yrityksessä tapahtuu ja miksi. Tarinat auttavat uusien ideoiden keksimistä: tarinankerronta toimii kanavana uusien ideoiden ja mielipiteiden jakamiseen yrityksen sisällä. Urheilutaustan merkitys korostuu Suunnossa myös konsepti- ja tuotekehityksessä sekä päätöksenteossa tuotekehitysprosessin aikana. Toimivat ratkaisut on helpompi erottaa, kun työntekijöillä on omakohtaista kokemusta tuotteen käyttökontekstista.

Erona perinteisiin lead-käyttäjiin, harrastelijat työskentelevät yrityksessä enneminkin vuosikausien kuin yksittäisten projektien ajan. Harrastelijatiedon heikkoutena voidaan nähdä, että sen avulla saatu epäsuora tieto on vaikea realisoida joksikin konkreettiseksi asiaksi, palveluksi tai tuotteeksi [Kot07].

2.1.5 Innovaatioprosessi

Lawrence M. Millerin kehittämällä *innovaatioprosessilla* tarkoitetaan korkeantason toimintasuunnitelmaa, jonka avulla yrityksen toimintakulttuuria pyritään saamaan innovatiivisemmaksi. Innovaatioprosessi koostuu neljästä vaiheesta: *löytämisestä* (discover), *unelmasta* (dream), *suunnittelusta* (design) ja *kehittämisestä* (develop) [Mil09].

Ennen kuin yrityksen tulevaisuutta on mahdollista suunnitella, on oltava ideoita, vaihtoehtoja ja mittareita päätöksien tueksi. Tämän vaiheen tulokset voidaan jakaa kahtia: ulkoisiin ja sisäisiin löytöihin [Mil09].

Ulkoiset löydöt ovat organisaation ulkopuolelta tulevia vaikutteita jotka voivat tuottaa uusia ideoita tai muuttaa organisaation toimintaan. Yrityksen toimintaympäristö on olennainen ulkoisten löytöjen tuottaja: markkinoiden, uusien teknologioiden ja sosiaalisen ympäristön muutokset tuottavat ulkoisia löytöjä.

Yrityksen sisäinen ympäristö vaatii arvojen, tehtävien, visioiden ja yhteisen strategian määrittämisen. Määritellyt periaatteet toimivat suunnittelutyön pohjana.

Olennaista on kuvata yrityksen ydinprosessi, esimerkiksi graafisia malleja hyödyntäen. Graafisesti kuvattu ydinprosessi auttaa löytämään organisaation vahvuudet. Myös tiimien tai yksilöiden sankarilliset saavutukset huomioidaan. Saavutuksien rooli korostuu, kun unelmaa tulevaisuuden organisaatiosta aletaan rakentaa. Nykyiset ydinprosessin mahdollistavat prosessit tunnistetaan. Sisäisen ympäristön vahvuuksien löytämiseen voi käyttää henkilökohtaisia haastatteluja, pienryhmätyöskentelyä tai laajemman mittakaavan konferensseja. Haastatteluissa kysytään ensiksi vahvuuksia ja positiivista suorituskykyä kartoittavia kysymyksiä ja sen jälkeen toiveita ja tarpeita esille tuovia kysymyksiä.

Unelma-vaiheessa vastataan kysymykseen: mitä voisimme olla, mitä emme vielä ole? Organisaation unelmien kehittämisessä auttavat kolme kysymystä:

1. Mikä olisi ideaali tulevaisuuden palvelu tai tuote asiakkaillemme? Miltä se näyttäisi, mitä se tekisi ja miltä se asiakkaista tuntuisi?
2. Mikä tekisi organisaatiostamme parhaan työpaikan tehtävän suorittamiseen? Miltä se tuntuisi? Millainen työasetelma kannustaisi ja kehittäisi organisaation työntekijöitä?
3. Kuinka kaksi aiempaa kysymystä tekevät organisaatiosta hyvän ja auttavat

saavuttamaan hyviä liiketoiminnallisia tuloksia?

Unelman kehittäminen kannattaa tehdä ryhmissä. Yhden työntekijän unelmat stimuloivat ideoita toisissa. Löytämis- ja unelmavaiheen lopputuloksena on muodostettava ”konsensusunelma”. Kaikista unelmista kehitetään yksi unelma, joka otetaan todelliseksi tavoitteeksi.

Kahdesta aiemmasta vaiheesta poiketen suunnittelu on käytännöllinen vaihe. Suunnittelu vastaa kysymykseen ”mitä todella on tehtävä, että unelmasta tulee totta?” Aiemmissa vaiheissa tuotetut ideat organisoidaan ja suunnitellaan loogisessa järjestyksessä. Lähtökohdaksi otetaan organisaation ydinprosessi. Ydinprosessi voidaan suunnitella asettumalla täysin uuden yrityksen saappaisiin: millainen yrityksen ideaali prosessi olisi, jos yrityksellä ei olisi mitään rajoitteita. Ydinprosessin mahdollistavat prosessit, kuten yrityksen käyttämä tietotekniikka tai henkilöstöosasto pitää suunnitella tukemaan ja optimoimaan ydinprosessia. Tämä vaatii olemassa olevien prosessien uudelleen suunnittelemista tai uusien vaatimusten määrittelyä olemassa oleville prosesseille.

Ydinprosessin suunnittelun jälkeen tuotetaan organisaatio, järjestelmät ja rakenteet prosessin ympärille. Organisaatio voidaan suunnitella esimerkiksi *alhaalta-ylös* (bottom-up). Organisaatiorakenteen pohja tehdään sen perusteella, että työ tulee (suurimmalla todennäköisyydellä) tehdyksi parhaalla mahdollisella tavalla: ensimmäiseksi määritellään ensimmäisen tason ryhmät, eli ydinalueen työntekijät. Tämän jälkeen pohditaan, mitä tukea ensimmäisen tason työntekijät tarvitsevat suoriutuakseen parhaalla mahdollisella tavalla päivittäisessä työssään. Lisäksi kaikki ydinprosessin kannalta olennaiset työtä tukevat järjestelmät on tunnistettava. Suunnitelmassa tuotetaan lista järjestelmään liittyvistä kysymyksistä ja ongelmista, jotka otetaan huomioon suunnittelutyössä.

Suunnitelma ei ole koskaan valmis – uuden prosessin käyttöönotto herättää parannusehdotuksia. Parannusehdotuksiin on varauduttava etukäteen ja työntekijöitä on kannustettava parannusehdotusten antamiseen. Suunnitelmaa on kyettävä päivittämään sitä mukaan, kun siihen löytyy parannettavaa.

2.2 Innovaatioita tukevat ohjelmistokehitysprosessit

Perinteisesti ohjelmistokehitysprosessit ovat olleet *suunnitelmakeskeisiä*.

Suunnitelmakeskeisyydellä tarkoitetaan sitä, että ohjelmistokehitys on perustunut tarkkaan etukäteiseen suunnitteluun ja raamittamiseen [Roy70, Rie11]. Nykyisin suunnitelmakeskeisestä ohjelmistokehityksestä puhutaan vesiputousmallina. Vesiputousmallin vaiheet perustuvat ajatukseen, jossa ohjelmisto kehitetään isoissa osissa [Roy70]. Prosessin ensimmäinen vaihe on *tarkka määrittely*, jonka aikana ohjelmisto ja sen toiminallisuus määritellään yksityiskohtaisesti. Toinen vaihe on *suunnittelu ja implementointi*, jonka aikana ohjelmiston yleisrakenne suunnitellaan ja ohjelmistokomponentit tunnistetaan. Tämän jälkeen järjestelmä implementoidaan käyttäen jotain ohjelmointikieltä, usein usean yksilön tai tiimin toimesta. Implementoinnin jälkeen toteutettu ohjelmisto tai ohjelmiston osa *integroidaan ja testataan*. Yksittäiset moduulit integroidaan olemassa olevaan järjestelmään ja testataan. Tämän jälkeen toteutetulla ohjelmistolla *operoidaan* ja sitä *ylläpidetään*. Ohjelmiston jaetaan asiakkaalle ja asiakkaan osoittamat muutostarpeet huomioidaan ja korjataan [Roy70]. Vaiheiden välillä voidaan liikkua edestakaisin, jolloin edeltävän vaiheen puutteellisuudet voidaan korjata ennen seuraavaan vaiheeseen siirtymistä. Suunnitelmakeskeisen prosessin ongelma on palautteen siirtyminen vaiheesta toiseen. Ongelmat määrittelyssä, suunnitelmassa ja implementoinnissa huomataan vasta kun järjestelmä on jo toteutettu.

Startup-yrityksen liiketoimintamalli asettaa ohjelmistokehitysprosessille erityisvaatimuksia. Startup-yritykset ovat nuoria yrityksiä, jotka vasta kehittävät ensimmäistä tuotettaan. Kaikki tuotteen vaatimukset eivät todennäköisesti ole selvillä tuotteen kehityksen alkaessa [Rie11]. Tästä johtuen suunnitelmakeskeiset – tarkkaa etukäteistä suunnittelua ja raamittamista vaativat – prosessimallit eivät sovi startup-yrityksen tarpeisiin [Rie11]. Ohjelmistoprosessin on siis oltava ketterä ja sopeutuva pikaisille muutoksille.

Ketterä ohjelmistoprosessi (Agile software process) on sellainen ohjelmistoprosessi, joka on ensisijaisesti nopea reagoimaan muutoksiin ja pyrkii minimoimaan kehitysprosessin aiheuttaman kuorman. Ketterä manifesti (Agile Manifesto) määrittelee ketterän kehityksen seuraavasti [BBB01]: ”*(Ohjelmistokehityksessä arvostamme) yksilöitä ja vuorovaikutusta enemmän kuin prosesseja ja työkaluja, toimivaa sovellusta enemmän kuin kokonaisvaltaista dokumentaatiota, asiakasyhteistyötä enemmän kuin sopimusneuvotteluita ja muutokseen reagoimista enemmän kuin suunnitelman noudattamista.*”

Seuraavat ominaisuudet ovat ketterille menetelmille tyypillisiä [Mil01, ASR02]:

1. Ohjelmakoodin jakaminen lyhyisiin ja yksinkertaisiin moduuleihin.
2. Iteratiivisuus, lyhyet syklit mahdollistavat nopeat verifioinnit ja korjaukset.
3. Yhdestä kuuteen viikkoa pitkät, aikaan sidotut syklit.
4. Nuukuus kehitysprosessissa poistaa kaikki turhat toimenpiteet.
5. Mukautuvuus mahdollisiin yllättäviin riskeihin.
6. Inkrementiaalinen lähestymistapa mahdollistaa järjestelmän rakentamisen pienissä osissa.
7. Yhtyvä ja inkrementiaalinen lähestymistapa minimoi riskit.
8. Ihmispainotteisuus.
9. Yhteistyökykyisyys ja kommunikoivat työtavat.

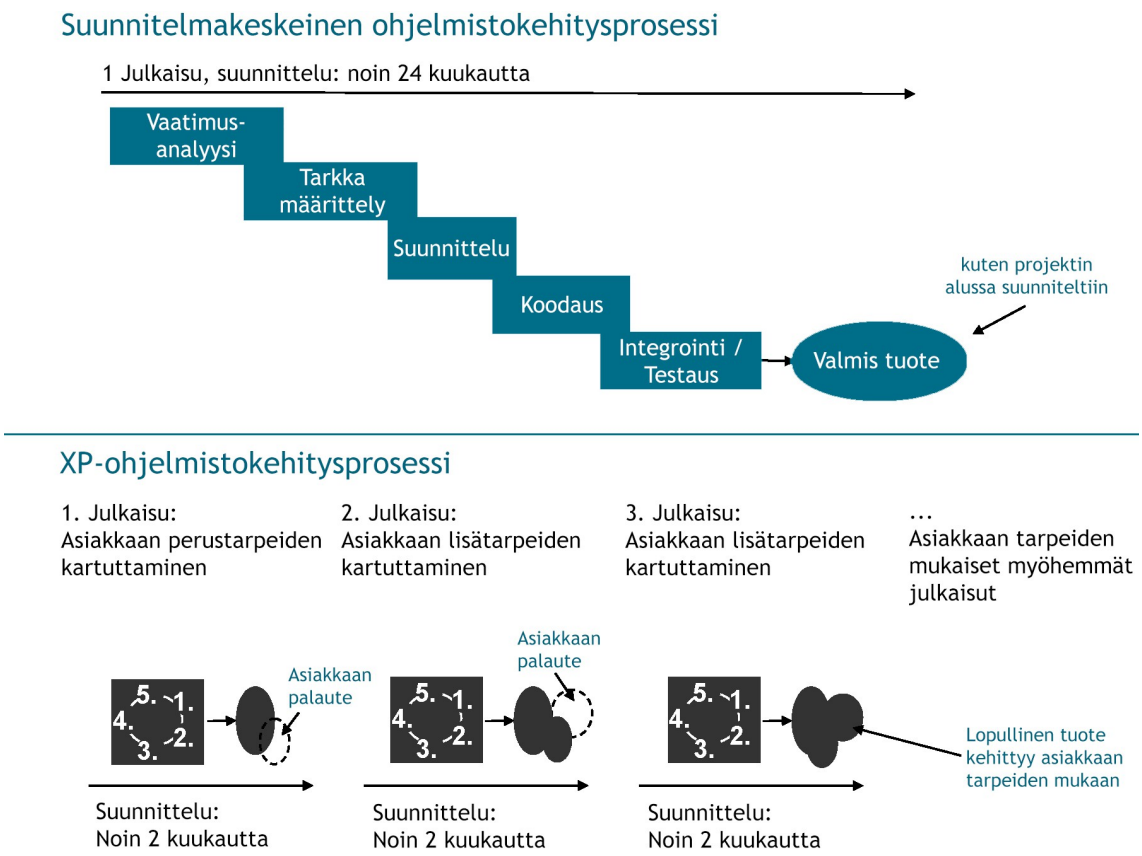
Uusien liiketoimintamallien testaaminen vaatii ohjelmistokehitysprosessilta ketteryyttä, jotta liiketoimintamallin toimivuus voidaan mahdollisimman nopeasti joko vahvistaa tai kumota [Rie11]. Jotta ohjelmistokehitys pystyy tukemaan innovaatiovetoista ohjelmistokehitystä, on sen luonteenomaisesti kyettävä reagoimaan nopeasti muutoksiin. Tässä luvussa kaikki läpikäytävät ohjelmistokehitysprosessit ovat ketteriä prosessimalleja. Esiteltävät menetelmät ovat Extreme Programming (XP), Scrum, Kanban ja Feature Driven Development (FDD).

2.2.1 Extreme Programming (XP)

Kent Beck kehitti vuonna 1998 ohjelmistotekniikan nimeltä *Extreme Programming, XP*. XP:n tavoite oli löytää uusi lähestymistapa ohjelmistokehitykseen, joka yksinkertaistaisi olemassa olevia kehitysmenetelmiä, joihin ohjelmistokehittäjät ovat tottuneet [GSW06]: kehitys on ketterämpää kuin aikaisemmissa, vesiputousmallia noudattavissa menetelmissä [ASR02].

Myöhemmin XP:stä tuli yksi suosituimmista ketteristä menetelmistä. XP-menetelmä nähdään yleisesti ketterien menetelmien käyttöönoton käynnistäjänä sekä uusien ketterien menetelmien innoittajana [ASR02]. XP:n pyrkii olemaan ratkaisu ohjelmistokehitykselle ympäristöissä, joissa vaatimukset vaihtuvat nopeasti [GSW06]. Kehitys pohjautuu lyhyihin *sykleihin* (loops), joista jokainen tuottaa *julkaisun* (release). XP-ohjelmistokehitys on siis luonteeltaan iteratiivista ja inkrementaalista.

Kuva 2 havainnollistaa kuinka uusi tuote kehittyy yhdessä asiakkaan kanssa julkaisu julkaisulta [GSW06]. Suurin osa suunnittelusta tapahtuu kehityksen aikana. Ensimmäiseksi suunnitellaan yksinkertaisin mahdollinen ratkaisu, jota laajennetaan osa kerrallaan. Uuden tuotteen kehitysprosessin alussa asiakas kertoo perustarpeensa uudelle tuotteelle, joiden perusteella suunnitellaan ensimmäisen julkaisun sisältö. Kun kehitystiimi on toteuttanut perustarpeet, asiakkaalle esitellään julkaisukelpoinen tuote. Perustarpeiden tyydyttäminen voi kestää esimerkiksi kaksi kuukautta. Ensimmäisen julkaisun perusteella asiakas auttaa määrittelemään seuraavan ominaisuuden, *käyttäjakertomuksen* (user story) avulla. Käyttäjäkertomus kuvaa tuotteen käyttötarpeet peruskäyttäjän näkökulmasta ja valottaa asiakkaan seuraavia tarpeita. Ohjelmistokehittäjät parantavat käyttäjäkertomuksen perusteella tarvittavia ominaisuuksia, tekemällä muutoksia ja lisäyksiä seuraavaan julkaisuun. Sama menettelytapa toistuu uudestaan ja uudestaan. Jokaisen syklin aikana aika, kustannus, laatu ja laajuus määritellään uudelleen. Lyhyet syklit mahdollistavat välittömän palautteen keräämistä asiakkaalta sekä suunnitelmien muutokset kesken ohjelmistokehitysprosessin. Tiheä julkaisutahti aiheuttaa sen, että vaatimusmäärittelyä, ohjelmointia ja testausta ei ole mahdollista tehdä suunnitelmakeskeisen kehityksen tapaan järjestyksessä, vaan ohjelmisto pikemminkin kehittyy yritys- ja oppimisprosessin aikana [GSW06].



Kuva 2: Suunnitelmakeskeinen ohjelmistokehitys verrattuna XP-ohjelmistokehitykseen [GSW06].

XP pitää sisällään 12 *toimintatapaa* (practice), joiden avulla hieman kaoottiseen ohjelmistoprosessiin saadaan kuria [Bec00].

1. *Suunnittelupeli* (planning game). Pelin ideana on hahmotella seuraavan julkaisun laajuus yhdistelemällä teknisiä arvioita ja liiketoiminnallisia prioriteetteja.
2. *Pienet julkaisut* (small releases). Tarkoituksena on saada pieni järjestelmä tuotantoon nopeasti ja päivittää siitä uusia versiota lyhyellä aikavälillä.
3. *Metaforan* (metaphor) tarkoitus on ohjeistaa kaikkea kehitystä yksinkertaisella jaetulla tarinalla siitä, kuinka koko järjestelmä toimii.
4. *Suunnittelun yksinkertaisuus* (simplicity in design) on tavoite. Järjestelmä on rakennettava niin yksinkertaiseksi kuin mahdollista. Monimutkaisuutta poistetaan heti kun sellaista havaitaan.
5. *Testaus* (testing) on merkittävässä roolissa. Ohjelmoijat kirjoittavat jatkuvasti yksikkötestejä. Kehitystä ei jatketa, ennen kuin yksikkötesti menevät läpi.
6. *Jatkuva integrointi* (continual integration). Aina kun tehtävä on valmis, se

integroidaan järjestelmään. Perusajatuksena on sisällyttää kehitettävät ominaisuudet järjestelmään niin aikaisessa vaiheessa kuin mahdollista.

7. *Pariohjelmointi* (pair programming). Kaikki tuotantoon päätyvä koodi kirjoitetaan kahden ohjelmoijan voimin, yhdellä koneella.
8. *Yhteisellä omistajuudella* (collective ownership) tarkoitetaan sitä, että kuka tahansa voi muuttaa järjestelmän mitä koodia tahansa milloin tahansa.
9. *Refaktorointia* (refactoring) toteutetaan jatkuvasti toiston poistamiseksi, kommunikoinnin parantamiseksi ja järjestelmän yksinkertaistamiseksi. Ohjelmoijat uudistavat järjestelmää muuttamatta sen käyttäytymistä.
10. Työnteon *tahti on oltava vakaa* (sustainable pace). Työntekijät eivät saa työskennellä enempää kuin 40 tuntia viikossa. Ylitöitä ei saa tehdä kahtena viikkona peräkkäin.
11. *Asiakkaan on oltava paikan* päällä (on-site customer) vastaamassa kysymyksiin.
12. *Yhteisten koodikäytäntöjen* (shared coding standards) noudattaminen, mahdollistaa, että ohjelmoijien tuottama koodi on laadukasta ja kaikille kehittäjille ymmärrettävää.

2.2.2 Scrum

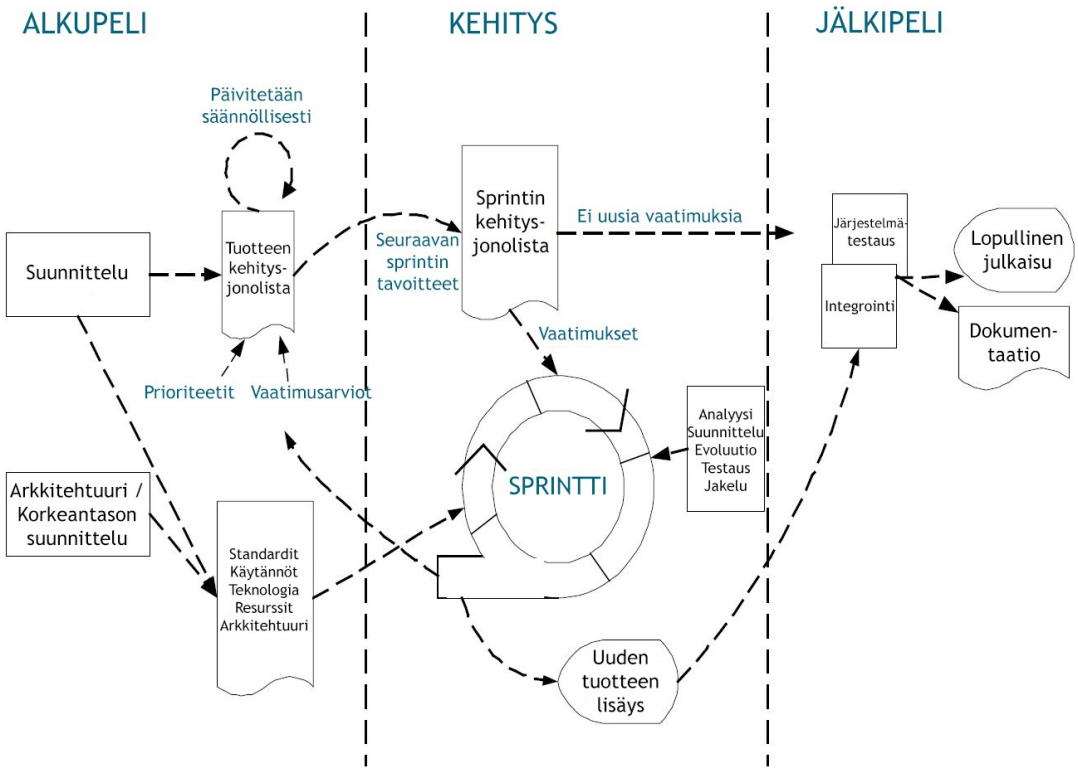
Scrum on yleisesti käytössä oleva ketterän ohjelmistokehityksen projektinhallinan viitekehys. Takeuchi ja Nokana kuvasivat Scrumin kaltaisen kehitysprosessin idean artikkelissaan *The New New Product Development Game* [TaN86]. Artikkelissa tuotekehitys kuvataan mukautuvana, nopeana ja itsestään organisoituvana kehitysprosessina. Vaikka Scrum onkin yleisesti käytössä ohjelmistokehityksessä, se ei määrittele yhtään ohjelmistokehitystekniikkaa, vaan on yleissopiva prosessimalli erilaisille projekteille [ASR02].

Scrumin prosessi sisältää kolme vaihetta [ASR02]: *kehitystä edeltävä vaihe* (pre-game), *kehityksen* (development) ja *kehityksen jälkeinen vaihe* (post-game).

Kehitystä edeltävä vaihe sisältää kaksi vaihetta: *suunnittelun* (planning) sekä arkkitehtuurin / korkean tason suunnittelun (High level design / Architechture). Suunnittelu määrittelee kehitettävän järjestelmän, jonka kiteytyy tuotteen *kehitysjonolistan* (backlog list) toteuttamisena. Kehitysjonolistaan kirjataan järjestelmään toteutettavat ominaisuudet pieniin osatehtäviin ositettuina. Kullekin ominaisuudelle määritellään prioriteetti sekä kirjataan ominaisuuden arvioitu työmäärä. Kehitysjonolistan vaatimukset voivat tulla asiakkaalta, myynti- tai markkinointiosastolta, asiakastuelta tai ohjelmistokehittäjiltä. Listaa päivitetään sitä mukaa, kun uusia vaatimuksia tulee, tai kun vaatimukset muuttuvat tai tarkentuvat. Suunnitteluun sisältyy myös projektitiimin, käytettävien työkalujen ja muiden resurssien, riskien ja koulutustarpeiden määritteleminen. *Scrumtiimi* (Scrum Team) tarkistaa päivitetyn kehitysjonolistan jokaisella iteraatiolla. Kehitysjonolistaan kirjattujen tehtävien perusteella suunnitellaan järjestelmän arkkitehtuuri korkealla tasolla. Sellaiset tehtävät tunnistetaan, jotka voivat vaatia muutoksia olemassa olevaan järjestelmän arkkitehtuuriin.

Kehitys on ketterä osa Scrum-prosessia [ScB02]. Vaihetta ajatellaan ikään kuin mustana laatikkona, jossa arvaamattomia asioita on odotettavissa. Eri ympäristö ja tekniset muuttujat kuten aika, kehys, laatu, vaatimukset, resurssit, teknologiat ja työkalut ja jopa kehitysmenetelmät – jotka voivat muuttua prosessin aikana – tunnistetaan. Scrum pyrkii mukautumaan muutoksiin joustavasti tunnistamalla tällaisia muuttujia myös kehityksen aikana. Kukin *sprintti* (Sprint) eli iteraatio kestää yhdestä viikosta yhteen kuukauteen. Järjestelmän kehitysprosessi voi sisältää esimerkiksi kahdeksan sprinttiä.

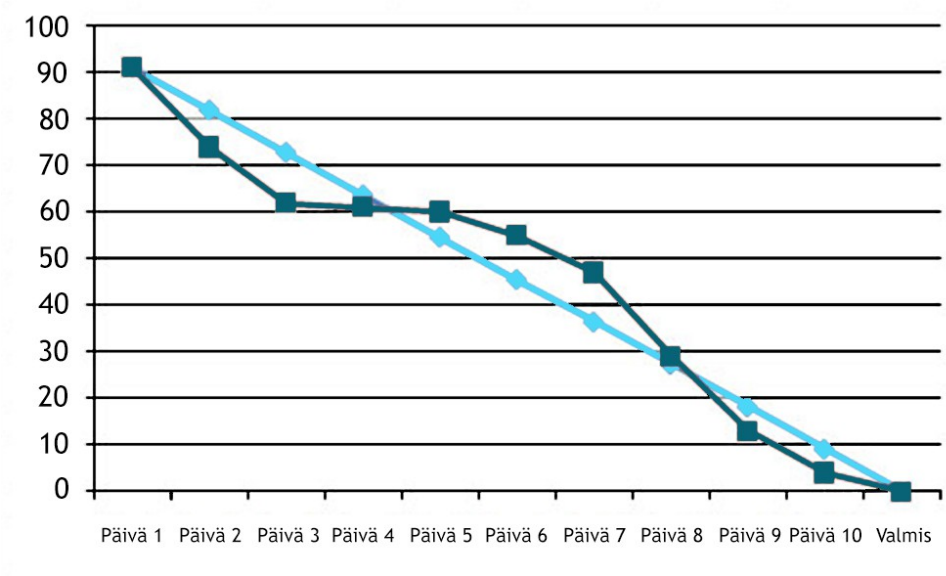
Kehityksen jälkeinen vaihe päättyy julkaisuun [ASR02]. Tähän vaiheeseen siirrytään, kun ympäristömuuttujat kuten vaatimukset on toteutettu. Kun kehityksen jälkeiseen vaiheeseen siirrytään, tehtäviä tai ongelmia ei enää ole eikä uusia keksitä. Julkaisukelpoisuus varmistetaan järjestelmätestauksella. Integrointi toteutetaan ja järjestelmä dokumentoidaan. Kehityksen jälkeisessä vaiheessa kehitettyjä ominaisuuksia tarkastellaan niille asetetun *valmiin määritelmää* (Defition of done) vasten. Valmiin määritelmä on kehitystiimin ja tuoteomistajan yhdessä sopima kriteeristö, jonka ominaisuuden on täytettävä.



Kuva 3: Scrum prosessi [ASR02].

Scrum sisältää 3 erilaista dokumentointimuotoa, jotka ovat toisiaan täydentäviä [ScB02, TaN86]. Tuotekehitysjonolista nähdään dokumenttina, joka pitää sisällään kaikki ominaisuudet. Tuotekehitysjonolistasta nähdään mitä aiotaan toteuttaa ja miten nopeasti sekä mitkä ovat tehtävien prioriteetit. Sprintin kehitysjonolista on dokumentti, joka luodaan suunnittelukokouksessa. Tähän dokumenttiin kerätään tehtävät, jotka tullaan sprintin aikana suorittamaan. Tehtävät voivat olla joko suunnittelutehtäviä,

ohjelmointitehtäviä, testaustehtäviä tai dokumentointitehtäviä. Tehtäviä ei osoiteta henkilöille, vaan kukin työntekijä ottaa yhden tehtävän tehtäväkseen. Arvio tehtävän valmistumisajasta päivitetään päivittäin. Sprintin kehitysjonolistaan ei voi lisätä tehtäviä kesken sprintin. Kolmas dokumentti on sprintin edistymiskäyrä (sprint burndown chart). Sprintin edistymiskäyrä kuvaa jäljellä olevat työtunnit koko sprintin työtaakan osalta suhteessa suunnitelmaan. Kuvasta 4 nähdään, kuinka edistyminen ja tavoitteellinen aikataulu on visualisoitu sprintin edistymiskäyrällä samaan diagrammiin. Suunnitelma on kuvattu kirkkaan sinisellä viivalla ja todellinen toteutuma tumman vihreällä viivalla. Diagrammi päivittyy joka päivä sitä mukaa, kun aikaa kuluu ja tehtävät edistyvät.



Kuva 4: Sprintin edistymiskäyrä.

Scrum sisältää kuusi roolia joilla on eri tehtävät ja vastuualueet prosessin aikana [ASR02]: *scrummaster* (Scrum Master), *tuoteomistaja* (Product Owner), scrumtiimi, *asiakas*, *käyttäjä* ja *johtajisto* (Management).

Scrummaster on vastuussa siitä, että projekti etenee sovittujen käytäntöjen sekä scrumin arvojen ja sääntöjen mukaan. Scrummaster on kanssakäymisessä projektitiimin, asiakkaan ja johtajiston kanssa projektin aikana. Scrummaster on myös vastuussa siitä, että tiimi pystyy työskentelemään esteittä mahdollisimman tehokkaasti.

Tuoteomistaja on virallisesti vastuussa projektista, projektin johtamisesta ja -hallinnasta. Scrummaster valitsee tuoteomistajan scrumtiimin jäsenistä. Tuoteomistaja tekee

viimeiset päätökset koskien kehitysjonolistan tehtäviä.

Scrumtiimi on projektitiimi, jolla on oikeus päättää tarvittavista toimenpiteistä ja organisoimaan rivinsä saavuttaakseen jokaisen sprintin tavoitteet. Scrumtiimi osallistuu ominaisuuksien työmäärän arviointiin, kehitysjonolistan luomiseen sekä arviointiin.

Johtajisto vastaa viime kädessä päätöksistä ja osallistuu tavoitteiden ja vaatimusten asetteluun. Johtajisto voi esimerkiksi olla mukana valitsemassa tuoteomistajaa, mittaamassa etenemistä ja supistamassa kehitysjonolistan tehtäviä.

Scrumissa on XP:n tapaan tiettyjä toimintatapoja [ScB02]. Kehitysjonolistan, listan tehtävien työmäärän arvioinnin ja sprinttien lisäksi Scrum pitää sisällään *sprintkohtaisia suunnittelukokouksia* (Sprint Planning Meeting), *sprintkohtaisia kehitysjonolistoja* (Sprint Backlog), *päivittäisiä scrum-tapaamisia* (Daily Scrum Meeting), *retrospektiivejä* (Retrospective) sekä *sprinttien katselmointitapaamisissa* (Sprint Review Meeting).

2.2.3 Lean ohjelmistokehitys

Lean ajattelun periaatteita on otettu käyttöön myös ohjelmistoteollisuudessa [JyR12]. Yksi Lean kehityksen pääpiirteistä on kaikenlaisen turhuuden eliminoiminen käytettävästä prosessista. Eräs Leanin työkaluista on *Kanban* tuotanto-operaatioiden hallintamenetelmä [IKN10]. Kanban luotiin osana Toyotan tuotantojärjestelmää, jonka tavoitteena oli maksimoida varaston kiertonopeus sekä minimoida työtehtäviin kulutettava aikaa säilyttäen jatkuva tuotantovirta. Kanban on signaaliväline, joka toimii luomalla ja liikuttamalla tuontantojärjestelmän osia käyttäen *veto-mekanismiä* (pull) [Hir08]. Veto-mekanismilla tarkoitetaan, että yrityksen prosessitoiminnot laukaistaan tarve-signaaleihin perustuen – tällöin prosessit tuottavat ainoastaan silloin, kun asiakkaat tai jokin toinen prosessi tarvitsevat niitä. Käytännössä Toytotan tuotantojärjestelmässä tämä ilmenee siten, että autonosan tuottaminen aloitetaan vasta sitten, kun jokin muu autonosa osoittaa tarpeen kyseistä osaa kohtaan.

Kanban sisältää toimintaperiaatteita, joita ovat [Kni10]:

1. Työnkulku on visualisoitava.
2. Kesken olevien työtehtävien määrä on rajoitettava.
3. Yhden tehtävän keskimääräistä läpimenoaikaa (Lead-time) on mitattava ja prosessi on optimoitava siten, että läpimenoaika on mahdollisimman lyhyt ja

ennustettava.

4. Toimintatavat on oltava yhtenäiset.
5. Toimintatapoja kehitetään jatkuvasti eri mallien ja sidosryhmien kanssa.

Ohjelmistoteollisuudessa työnkulun visualisointi toteutetaan usein Kanban-taulua käyttäen [Hir08]. Kanban-taulu voidaan sijoittaa esimerkiksi työskentelytilan seinälle. Kuvasta 5 nähdään, kuinka tauluun lisätään järjestelmän eri työvaiheet nimetyin sarakkein sekä työvaiheiden sisällä olevat tehtävät. Tehtävät voidaan merkata tauluun erivärisin kortein, esimerkiksi muistilappujen avulla. Työnkulku etenee veto-mekanismia käyttäen, jolloin uutta työtä ei oteta kehitykseen ennen kuin tehtävälle on tilaus ja edellinen tehtävä on saatu valmiiksi. Kanbanin toinen periaate, työtehtävien määrän rajoittaminen näkyy myös Kanban-taululla. Kussakin sarakkeessa voi olla esimerkiksi maksimissaan kaksi tehtävää yhtä työntekijää kohden. Rajoitukset voidaan merkata numeroin kunkin sarakkeen kohdalle. Työtehtävät liikkuvat taululla vasemmalta oikealle sitä mukaan, kun ne täyttävät kaikki sen hetkisen tilansa kriteerit – valmiiksi saatu työtehtävä siirretään seuraavaan sarakkeeseen [Kni10].

Vaikka Kanban on alunperin toteutettu autoteollisuuden tarpeisiin, se on soveltuva myös ketterän ohjelmistoteollisuuden tarpeisiin [Hir08]. Ketterä ohjelmistokehitys perustuu tuotteen inkrementaaliseen tuottamiseen, jossa tuote rakennetaan julkaisu kerrallaan, lisäten edelliseen julkaisuun uutta toiminnallisuutta. Ketterän Kanban-taulun mukainen prosessimalli perustuu iteraatioihin [Hir08]. Kunkin iteraation alussa kehitettävät ominaisuudet hahmotellaan esimerkiksi käyttäjäkertomusten avulla, jotka kuvaavat tulevien ominaisuuksien tarpeet ja käyttötilanteet. Kuvasta 5 nähdään, kuinka Kanban-taulu kuvaa kehitysprosessin yhden iteraation vaiheet. Ensimmäinen sarake kuvaa kehitysjonolistaa, johon käyttäjäkertomukset on paloiteltu yksittäisiksi tehtäviksi. Toinen sarake kuvaa kehityksessä olevia tehtäviä, jotka on poimittu kehitysjonolistasta. Kolmas sarake kuvaa testattavina olevia tehtäviä, jotka vastaavasti on poimittu edellisestä sarakkeesta, niiden toteutuksen valmistuttua. Kolmas sarake kuvaa julkaisua odottavia tehtäviä ja viimeinen sarake tehtäviä, jotka on jo siirretty tuotantoon. Sulkujen sisässä olevat numerot kuvaavat montako tehtävää kussakin sarakkeessa voi olla kesken. Kirjaimet A-K kuvaavat työtehtäviä.

Kehitysjonolista (5)	Kehityksessä (3)	Testattavana (2)	Odottaa julkaisua (3)	Tuotannossa
H *I* *J* *K*	*F* *G*	*D* *E*	*C*	*A* *B*

Kuva 5: Ketterän Kanbanin työnkulku visualisoituna Kanban-taulun avulla [Kni10].

Yksi muistilappu Kanban-taululla on jonkin tehtävän yksi osatehtävä. Muistilappu voi sisältää tiedon esimerkiksi tehtävän nimestä, tehtävätunnisteesta, aika-arviosta ja tekijäksi osoitetun henkilön nimestä [Hri08].

Kanbanin hyviä puolia ovat [Kni10]:

- Mahdolliset pullonkaulat näkyvät reaaliajassa. Tämä edesauttaa työntekijöiden yhteistyötä arvoketjun optimoimiseksi.
- Kanban vie organisaatiota asteittain kohti ketteryyttä. Yritykset, jotka ei aiemmin ole olleet kykeneviä tai halukkaita koettamaan ketteriä menetelmiä, siirtyvät kohti ketteryyttä.
- Mahdollistaa ketterän ohjelmistokehityksen tarvittaessa myös ilman iteraatiota.
- Kanbanilla on tapana luonnollisesti levitä muihin osastoihin organisaation sisällä, mikä lisää yrityksen läpinäkyvyyttä.

2.2.4 Feature Driven Development (FDD)

Feature Driven Development (FDD) on ketterä ja adaptiivinen lähestymistapa järjestelmien kehittämiseen. FDD pyrkii yhdistämään iteratiivisen ohjelmistokehityksen parhaisiin – teollisuudessa tehokkaiksi osoittautuneisiin – toimintatapoihin. Menetelmä painottaa laadun merkitystä läpi kehitysprosessin [ASR02]. FDD keskittyy suunnittelu ja implementointivaiheisiin, eikä se sisällä kokonaista ohjelmistokehitysprosessia. FDD on kuitenkin suunniteltu toimimaan yhdessä muiden ohjelmistokehitysprosessien aktiviteettien kanssa [PaF02]. FDD ei täten vaadi, että sen yhteydessä tulisi käyttää jotain tiettyä kehitysprosessia. FDD painottaa useita ja todellisia julkaisuja sekä tarkkaa projektin kehityksen monitorointia [ASR02].

FDD koostuu viidestä peräkkäin suoritettavasta prosessista, sekä se tarjoaa menetelmät, tekniikat ja sidosryhmien tarpeet tyydyttävät ohjenuorat [ASR02]. Lisäksi FDD sisältää projektin tarvitsemat roolit, tavoitteet ja aikataulutuksen. Toisin kuin jotkut muut ketterät menetelmät, FDD pyrkii olemaan soveltuva toiminnaltaan kriittisten järjestelmien toteuttamiseen [PaF02].

FDD jakaa projektiin osallistuvien työntekijöiden roolit kolmeen kategoriaan: avainroolit (Key roles), tukevat roolit (Supporting roles) ja lisäroolit (Additional roles) [PaF02]. Kuusi avainroolia ovat [PaF02, ASR02]:

1. *Projektipäällikkö* (Project manager) on projektista ja taloudesta vastaava projektin johtaja. Yksi hänen tehtävistään on minimoida projektitiimin ulkoiset häiriötekijät ja mahdollistaa tiimille hyvät työskentelyolosuhteet. FDD:ssä projektipäälliköllä on viime kädessä sanavalta projektin laajuuteen, aikatauluun ja henkilövalintoihin.
2. *Pääarkkitehti* (Chief architect) on vastuussa järjestelmän kokonaissuunnitelmasta sekä suunnittelutiimin yhteisten suunnitelmakokousten järjestämisestä. Pääarkkitehti päättää viime kädessä kaikista suunnittelukysymyksistä.
3. *Kehityspäällikkö* (Developent manager) johtaa päivittäisiä kehitystoimenpiteitä ja ratkaisee mahdolliset tiimin sisäiset konfliktit. Lisäksi kehityspäällikön rooliin sisältyy resurssiongelmien ratkaiseminen.
4. *Pääohjelmoija* (Chief programmer) on kokenut kehittäjä, joka osallistuu vaatimusanalyysiin ja projektin suunnitteluun. Pääohjelmoija on vastuussa uusia

ominaisuuksia analysoivista, suunnittelevista ja toteuttavista kehitystiimeistä. Pääohjelmoija myös vastaa ominaisuusjoukosta kehitykseen otettavien ominaisuuksien sekä valituille ominaisuuksille sopivien luokkien omistajien valitsemisesta. Pääohjelmoijia voi olla useampia samassa projektissa.

5. *Luokan omistaja* (Class owner) työskentelee pääohjelmoijan opastuksessa tehden suunnittelua, ohjelmointia, testausta ja dokumentointia. Hän on vastuussa hänelle osoitetun luokan kehittämisestä. Luokan omistajat muodostavat yhdessä ominaisuustiimejä.
6. *Sovellusasiantuntijat* (Domain experts) voi olla käyttäjä, asiakas, sponsori, liiketoiminta-analyytikko tai jokin näiden yhdistelmä. Hänen tehtävänsä on tarjota tietoa siitä, kuinka erilaiset järjestelmävaatimukset tulisi toteuttaa. Sovellusasiantuntia jakaa tietonsa kehittäjille varmistaen, että kehittäjät toteuttavat oikeanlaisen järjestelmän.

Viisi tukevaa roolia ovat:

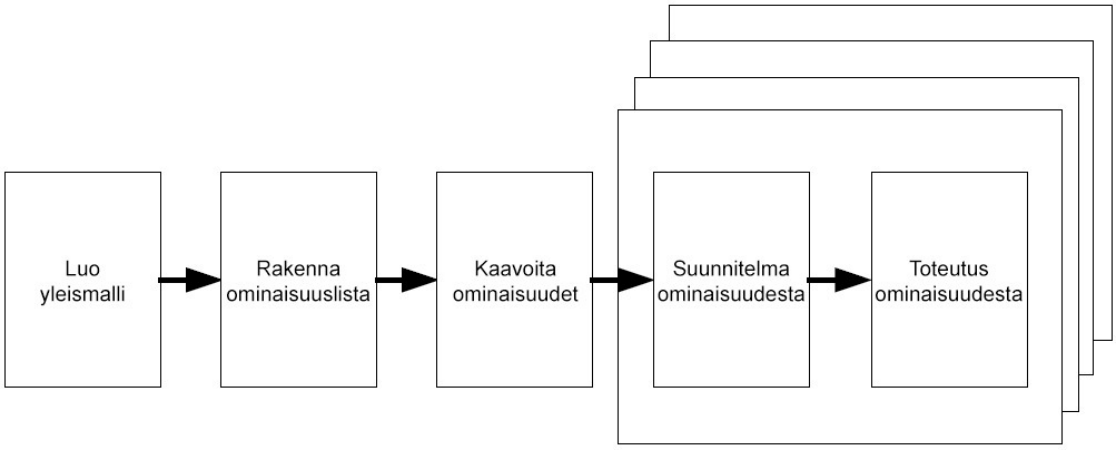
1. *Julkaisupäällikkö* (Release manager) kontrolloi prosessin kehittymistä tarkastelemalla pääohjelmoijien tuottamia raportteja ja pitämällä kokouksia pääohjelmoijien kanssa. Julkaisupäällikkö raportoi projektipäällikölle.
2. *Kieliguru* (Language lawyer/Language guru) on tiimin jäsen, joka on vastuussa perusteellisen tiedon tarjoamisesta esimerkiksi jonkin tietyn ohjelmointikielen tai teknologian osalta. Rooli on erityisen tärkeä silloin, kun projektitiimi työskentelee jonkin uuden teknologian parissa.
3. *Tuotantoonvienti-insinööri* (Build engineer) on vastuussa tuotantoonvientiin liittyvistä toimenpiteistä, kuten ympäristön pystyttämisestä, ylläpitämisestä ja tuotantoon viemisestä. Hän on vastuussa versiohallinan hallitsemista sekä dokumentaation julkaisemista.
4. *Työkaluseppä* (Toolsmith) on rooli pienten työkalujen rakentamiseen projektin kehitystä, testausta ja datakonversioita varten. Hän voi myös työskennellä tietokantojen luomis- ja ylläpitotehtävissä sekä projektikohtaisten verkkosivujen parissa.
5. *Järjestelmävastaava* (System administrator) konfiguroi, hallitsee ja selvittää

palvelinten, sisäverkon sekä kehitys- ja testausympäristöjen ongelmat.

Kolme lisäroolia ovat:

1. *Testaajat* (The Testers) varmistavat, että kehitettävä järjestelmä täyttää asiakkaan asettamat vaatimukset.
2. *Käyttöönottajat* (Deployers) konvertoivat olemassa olevaa dataa uuden järjestelmän vaatimaan muotoon ja osallistuvat uusien julkaisujen käyttöönottoon.
3. *Tekniset kirjoittajat* (Technical writers) toteuttavat käyttäjille suunnatun dokumentaation.

FDD-prosessin koostuu viidestä vaiheesta: *luo yleismalli* (Develop an Overall Model), *rakenna ominaisuuslista* (Build a Features List), *kaavoita ominaisuudet* (Plan by Feature), *suunnitelma ominaisuudesta* (Design by Feature) sekä *toteutus ominaisuudesta* (Build by Feature) [PaF02]. Kuvasta 6 on huomattavissa, kuinka viiden peräkkäin suoritetun prosessin aikana järjestelmä suunnitellaan ja implementoidaan.



Kuva 6: FDD-prosessi [ASR02, PaF02].

FDD-prosessin iteratiivinen osuus sisältää suunnittele ja toteuta -vaiheet, mahdollistaen nopean reagoimisen viime hetken vaatimuksiin ja liiketoiminnallisiin muutoksiin. Tyypillisesti ominaisuuden iteroiminen vie kehitystiimiltä aikaa yhdestä kolmeen

viikkoa.

Seuraavassa käydään prosessiin vaiheet läpi yksityiskohtaisemmin.

Luo yleismalli:

Kun yleismalli luodaan, sovellusasiantuntijat esittävät korkean tason kuvauksen järjestelmästä projektitiimin jäsenille sekä pääarkkitehdille. Tässä vaiheessa sovellusasiantuntijoilla on jo tiedossa tulevan järjestelmän laajuus, konteksti, järjestelmävaatimukset sekä dokumentoidut vaatimukset, kuten käyttäjäkertomukset [PaF02]. FDD ei kuitenkaan itsessään sisällä mallia vaatimusten kartoittamiseen [ASR02].

Luotu yleismalli jaetaan edelleen pienempiin osa-alueisiin. Kullekin osa-alueelle esitetään samankaltainen korkeantason kuvaus. Kunkin kuvauksen perusteella kehitystiimi luo luokkia esitetystä alueesta. Luoduista luokista keskustellaan ja niistä valitaan sopivimmat. Tämän prosessin aikana yleinen malli järjestelmästä muodostuu [PaF02].

Rakenna ominaisuuslista:

Korkeantason kuvaukset, oliomallit ja olemassa oleva vaatimusdokumentaatio luovat pohjan kattavan ominaisuuslistan muodostamiselle toteutettavasta järjestelmästä [PaF02]. Ominaisuuslistassa kehitystiimi esittää kaikki asiakkaalle arvoa tuottavat toiminnot (Client valued functions). Toiminnot esitetään erikseen jokaiselle osa-alueelle muodostaen joukko tärkeimmistä ominaisuuksista. Tärkeimmän ominaisuudet jaetaan edelleen yksittäisiin ominaisuuksiin, jotka kuvaavat eri toiminnot järjestelmän osa-alueen sisällä. Käyttäjät ja sponsorit käyvät vielä ominaisuuslistan läpi tarkastaen listan kelvollisuuden ja täydellisyyden.

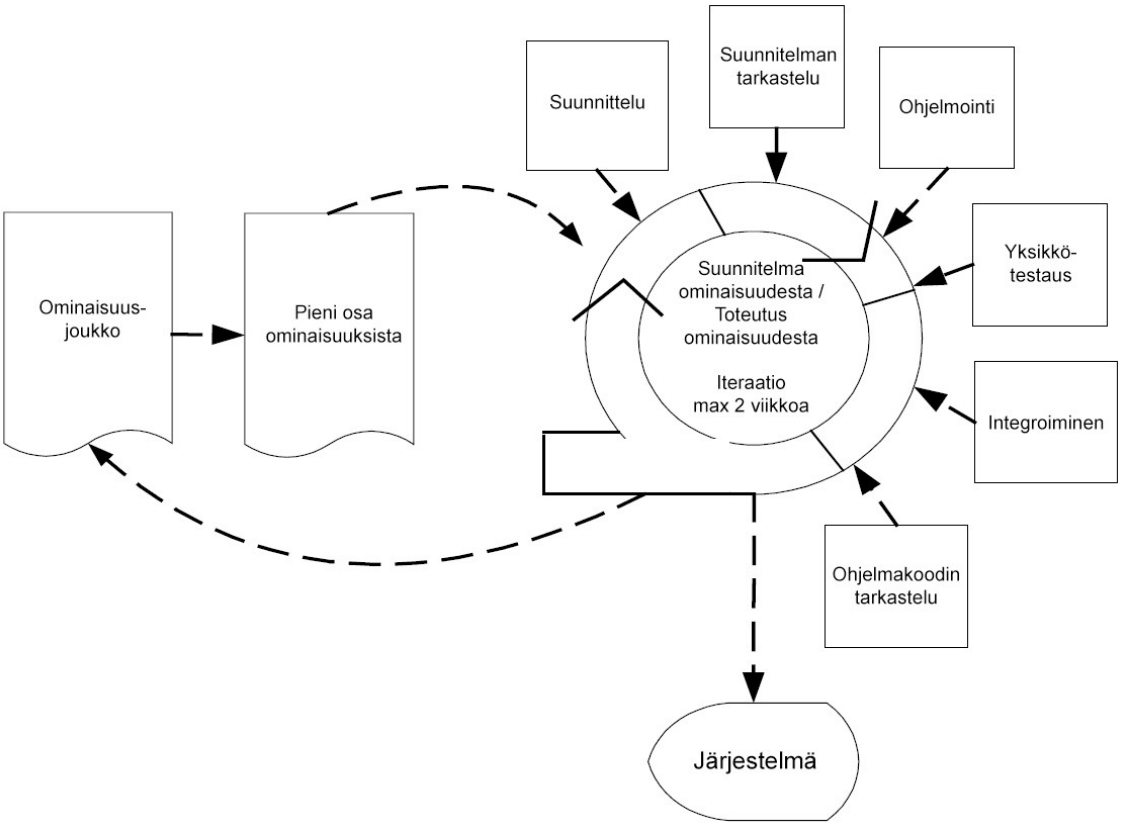
Kaavoita ominaisuudet:

Ominaisuuksien kaavoittaminen tarkoittaa korkeantason suunnitelman laatimista, jossa ominaisuusjoukon ominaisuudet järjestetään prioriteetin sekä mahdollisten riippuvuussuhteiden perusteella ja osoitetaan pääohjelmoijille [PaF02]. Lisäksi yleismallin luonnin yhteydessä muodostetut luokat osoitetaan yksittäisille kehittäjille, joista tulee luokanomistajia.

Suunnitelma ominaisuudesta (ja toteutus ominaisuudesta):

Ominaisuusjoukon ominaisuuksista valitaan pieni osa, joita aletaan toteuttamaan [PaF02]. Ominaisuustiimit muodostetaan asiaankuuluvista luokanomistajista. Suunnitelma ominaisuudesta ja toteutus ominaisuudesta -vaiheet muodostavat yhdessä iteratiivisen prosessin, jonka aikana ominaisuus implementoidaan. FDD pyrkii tiheään julkaisutahtiin, jolloin yhden iteraation tulee kestää korkeintaan kaksi viikkoa. Ominaisuustiimejä voi olla useampia, jotka työskentelevät yhtä aikaa omien ominaisuuksiensa parissa.

Kuvasta 7 huomataan, että iteratiivinen prosessi pitää sisällään suunnittelun lisäksi ohjelmoinnin, yksikkötestauksen, integraation ohjelmakoodin katselmoinnin sekä suunnitelman katselmoinnin [PaF02].



Kuva 7: Iteratiivinen suunnitelma ominaisuudesta ja toteutus ominaisuudesta -prosessi [ASR02].

FDD sisältää XP:n tavoin joukon suositeltavia toimintatapoja. Seuraavaksi esiteltävistä toimintatavoista kaikkia tulisi käyttää, saadakseen FDD-menetelmästä saa parhaan mahdollisen hyödyn [PaF02]. FDD sisältää seuraavat toimintatavat [ASR02]:

- *Kohdealueen oliomallintamisella* (Domain Object Modelling) tarkoitetaan kohdealueen tutkimista ja sen selittämistä. Mallintaminen tuottaa

lopputuloksenaan kehyksen, johon ominaisuudet lisätään.

- *Ominaisuuksien kautta toteuttaminen* (Developing by Feature). Kehitys toteutetaan toiminnallisuuslistan avulla, joka pitää sisällään kehitettävät ominaisuudet pieniin osiin hajoitettuna.
- *Yksilöllinen luokanomistajuus* (Individual Class Ownership) tarkoittaa sitä, että kullakin luokalla on omistaja, joka vastaa luokan yhtenäisyydestä, tehokkuudesta ja oikeanlaisuudesta.
- *Ominaisuustiimit* (Feature Teams) ovat pieniä saman osa-alueen parissa työskentelevistä työntekijöistä koostettuja tiimejä.
- *Tarkastukset* (Inspections) toteutetaan parhailla olemassa olevilla katselmointimenetelmillä.
- *Säännölliset tuotantoonviennit* (Regular builds).
- *Konfiguraatiohallinta* (Configuration Management) mahdollistaa viimeisimpien versioiden tunnistamisen ja vanhojen lähdekooditiedostojen hallinnan.
- *Etenemisraportin* (Progress reporting) avulla raportoidaan tehty työ kaikilla tarpeellisilla organisaatiotasoilla.

2.3 Startup-yritys

Startup-yritys on yleensä nuori nopeaa kasvua tavoitteleva ja skaalautuvalla liiketoimintamallilla operoiva yritys. Kasvuyritys on millä tahansa alalla toimiva yritys, joka operoi kasvua tukevalla liiketoimintamallilla. Startup-yritys on siis kasvuyritys, joka on toimintansa alkuvaiheessa. Startup-yritykselle on luonteenomaista olla innovatiivinen toimija. Eric Ries määrittelee startup-yrityksen seuraavasti: "startup-yritys on ihmisinstituutio, joka on suunniteltu luomaan uusia tuotteita ja palveluita äärimmäisen epävarmoissa olosuhteissa". Tämä aliluku perustuu Eric Riesen The Lean Startup -kirjaan, ellei erikseen mainita.

Mikään liikeidea ei ole toimintavarma syntyessään, vaan yrityksen sen hetkinen suunnitelma asetetun tavoitteen saavuttamiseksi. Tästä johtuen, myös yrityksen liiketoimintamallin voidaan nähdä sisältävän ainoastaan joukon oletuksia tuotteestaan ja vallitsevasta toimintaympäristöstä. Hyvä suunnitelma, vahva strategia ja kattava markkinointitutkimus eivät ole startup-yritykselle onnistumisen takeita. Kasvuyrityksen

prosessien pitää olla sellaisia, että ne kykenevät lyhyellä aikajänteellä vahvistamaan oletukset joko oikeiksi tai vääriksi.

Ries on kehittänyt *Lean Startup* –menetelmän, joka mahdollistaa markkinointioletusten nopean testaamisen prototyypeillä, ja hyödyntää asiakkailta saatua palautetta vahvistaakseen esitettyjä oletuksia. Lean startup on uusi innovatiivisten tuotteiden kehitysmenetelmä joka korostaa nopeita iteraatioita, asiakkailta saatavaa tietoa, suurta visiota ja korkeiden tavoitteiden asettamista. Menetelmä rohkaisee tavoittelemaan suuria ja aloittamaan pienestä. Menetelmän ydin on toteuttaa mahdollisimman nopeasti oikeille asiakkaille suppein mahdollinen toimiva tuote, jonka avulla tuotteen toimivuudesta voidaan *oppia* projektin aikaisessa vaiheessa.

Lean on *turhuuden* (waste) karsimiseen, joustavuuteen ja maksimaalisen arvon tuottamiseen pyrkivä johtamisfilosofia [Lik04]. Lean on myös avoin muutokselle. *Lean ajattelussa* (Lean Thinking) kaikki mikä ei lisää asiakkaan tuotteesta saamaa arvoa, on turhaa. Womack ja Jones kuvaa Lean ajattelun viisi ydinkonseptia [WJR90]:

1. asiakas määrittää arvon,
2. arvoketju on tunnistettava ja kaikki mikä ei lisää asiakkaan tuotteesta saamaa arvoa on poistettava,
3. arvoketju pohjautuu asiakkaan tarpeisiin perustuvaan imuohjaukseen,
4. kaikki yrityksen työtekijät osallistuvat tuotekehitykseen ja
5. toimintaa on kehitettävä jatkuvasti.

Lean Startup yhdistää Lean ajattelun ideat yrittäjyyteen. Lean Startupin pyrkimyksenä on olla kattava teoria, joka pystyy vastaamaan yrityksen eri vaiheiden, kuten visioiden ja konseptin, tuotekehityksen, markkinoinnin ja myynnin, skaalautuvuuden, kumppanuussuhteiden ja jakelun sekä organisaatiosuunnittelun tuomiin haasteisiin. Lean Startup menetelmän pyrkii luomaan yrityksen toimintaan kehikon, jonka avulla arvaukset ja oletukset ovat testattavissa.

Lean Startup -menetelmä sisältää viisi periaatetta, jotka ovat:

1. *yrittäjyyttä on kaikkialla* (entrepreneurs are everywhere),
2. *yrittäjyys on johtamista* (entrepreneurship is management),
3. *vahvistettu oppiminen* (validated learning),

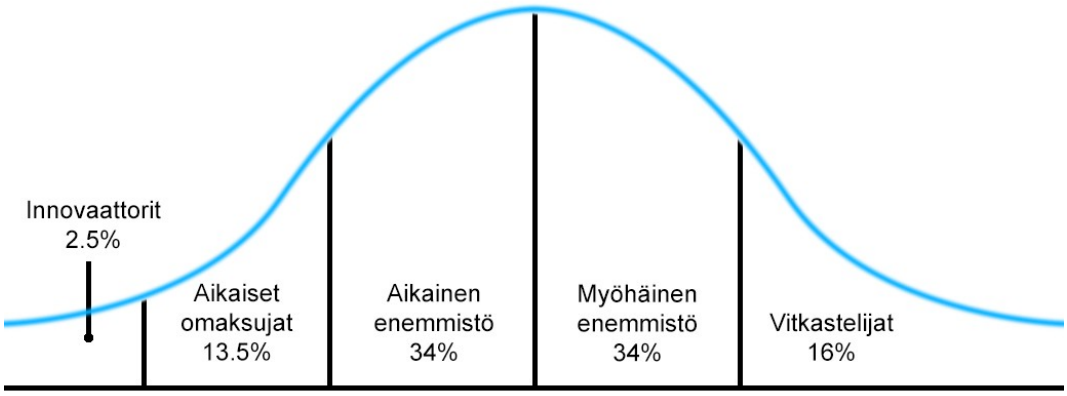
4. *luo-mittaa-opi* (build-measure-learn),
5. *innovaatiokirjanpito* (innovation accounting),

Ensimmäisellä periaatteella tarkoitetaan sitä, että Lean Startup -menetelmää voi toteuttaa minkä kokoisessa yrityksessä tahansa ja millä tahansa sektorilla tai teollisuuden alalla. Yrittäjällä tarkoitetaan sellaista henkilöä, joka työskentelee Riesin startup-määritelmän tapaisessa työyhteisössä. Lean Startup -menetelmä pätee siis niin uusiin yrittäjiin kuin yrityksen sisällä tapahtuvaan liiketoiminnan kehitykseenkin.

Toinen periaate tarkoittaa sitä, että startup-yritystä ei voida nähdä ainoastaan tuotteen kautta, vaan instituutiona. Startup-yritys tarvitsee uudenlaisen – epävarmoihin olosuhteisiin sopivan – johtamisnäkökulman. Riesin mukaan jokaisessa modernissa innovaatiovetoisessa yrityksessä tulisi olla työntekijä, joka toimii tehtävänimikkeellä *intrapreneur*. Monille yrittäjille käsitteet johtaminen ja yrittäjyys ovat ristiriidassa. Johtamisen nähdään aiheuttavan turhaa byrokratiaa ja tukahduttaa luovuutta. Tästä johtuen, useat startup-yritykset ovat sivuuttaneet johtamiselementit tyystin ja operoineet ”just do it”-asenteella. Tällaisella asenteella on taipumus johtaa useammin kaaokseen kuin onnistumiseen. Startup-yrityksellä tulee olla kurinalainen prosessi ja hallittu toimintatapa visionsa saavuttamiseksi.

Startup-yritykset eivät siis ole olemassa ainoastaan valmistaakseen tuotteitta, tuottaakseen rahaa tai palvellakseen asiakkaitaan. Kolmannella periaatteella, vahvistetulla oppimisella, Ries tarkoittaa sitä, että startup-yritys on olemassa myös oppiakseen kuinka vakaata liiketoimintaa voidaan toteuttaa. Yrityksen on ymmärrettävä, mitä asiakkaat todellisuudessa haluavat, eikä mitä asiakkaat sanovat haluavansa. Vahvistettu oppiminen on menetelmä, joka pyrkii eliminoimaan turhat oppimiskokemukset, joissa pitkään toteutettu epäonnistunut strategia on johtanut epäonnistumiseen jonka ainoa positiivinen tulos on oppimiskokemus vallitsevista markkinoista. Vahvistettu oppiminen lähtee siitä, että strategia on mahdollista vahvistaa ennen kuin sen toteuttamiseen on käytetty runsaasti resursseja. Oppiminen voidaan vahvistaa tieteellisesti toteuttamalla useita kokeita, jotka mahdollistavat yrittäjien jokaisen vision jokaisen elementin testaamisen. Vahvistettu oppiminen pohjautuu täten empiiriseen dataan, joka on kerätty aidoilta asiakkailta. Vahvistettu oppiminen on konkreettisempaa, tarkempaa ja nopeampaa kuin markkinoiden ennustaminen. Vahvistettu oppiminen vaatii, että tuote viedään todelliseen toimintaympäristöönsä todellisten asiakkaiden käytettäväksi niin nopeasti, kuin mahdollista. Tällainen

minimaalinen toimivan tuote (Minimal Viable Product), MVP, mahdollistaa empiirisen datan keräämisen ja nopean oppimisen. MVP myös antaa suuntaa siitä, pystyykö tuotteen tai palveluiden ympärille perustamaan kestävää liiketoimintaa. Lean Startup -menetelmän mukaan yrityksen jokainen tuote, ominaisuus, markkinointikampanja – ylipäätään kaikki mitä yritys tekee – tulee toteuttaa vahvistetun oppimisen periaatteiden mukaisesti. Tällöin vahvistetun oppimisen avulla starup-yrityksen strategian suunta hahmottuu jo hyvin aikaisessa vaiheessa projektia. MVP suunnataan usein *aikaisille omaksujille* (early adopters). Kuvasta 8 nähdään, että aikaiset omaksujat ovat vain pieni osa – 13,5 prosenttia – koko käyttäjäkunnasta [Rog83]. Tuotteen tai palvelun ei tarvitse olla lähtökohtaisesti täydellinen, sillä aikaiset omaksujat ovat lähinnä kiinnostuneet tuotteen uusista ominaisuuksista, eikä perustoiminnallisuuden puutteet ole heille niin häiritseviä. Kun MVP on osoittanut suosiota aikaisten omaksujien parissa, tuotetta on luontevaa kaupata valtavirta-asiakkaille. On kuitenkin huomioitava, että valtavirta-asiakkaiden vaatimukset voivat olla erilaiset kuin aikaisilla omaksujilla.



Kuva 8: Innovaatioiden omaksumiskäyrä [Rog83].

Startup-yrityksen perimmäinen tarkoitus on muuttaa ideat tuotteiksi ja mitata kuinka asiakkaat reagoivat niihin. Reagoinnin perusteella opitaan, kannattaako tuotteen kehittämistä jatkaa. Luo-mittaa-opi –periaatteella toimiva palautesilmukka tulee sisällyttää jokaiseen startup-yrityksen käyttämään prosessiin. Silmukan ensimmäinen vaihe, *luo*, tarkoittaa pikaisen MVP:n toteuttamista. Toinen vaihe *mittaa* tarkoittaa MVP:n toimivuuden mittaamista ja *opi* tarkoittaa mitatusta toimivuudesta oppimista. Prosessi mahdollistaa yrityksen liiketoimintamallin, eli käytännössä yrityksen arvausten ja oletusten, systemaattisen testaamisen. Palautesilmukka varmistaa, että yritys tietää mihin suuntaan se on menossa.

Yrityksen strategian tuloksista pidetään innovaatiokirjanpitoa, joka on kvantitatiivinen menetelmä yrityksen innovaatioiden tulosten mittaamiseen. Innovaatiokirjanpidon tarkoitus on parantaa yrityksen tulosta ja pitää innovaattorit vastuuvelvollisina. Etenemistä mitataan, merkkipaaluja asetetaan ja työtä priorisoidaan. Innovaatiokirjanpito kostuu kolmesta vaiheesta:

1. Innovaatiokirjanpito vaatii, että yrityksen sen hetkinen tilanne kartoitetaan keräämällä dataa MVP:n avulla. Tilanne on kartoitettava, jotta yrityksen etenemistä voi seurata.
2. Startupin kehittää toimintaansa ja optimoi tuotteitaan kohti visiotaan. Tämä voi vaatia useita yrityksiä.
3. Toiminnan kehittäminen tai *muutos* (pivot). Jos yritys etenee hyvin kohti ideaalitilannettaan – ja onnistuu vahvistamaan oletuksiaan – toimintaa ei ole syytä muuttaa. Jos ei, nykyinen strategia on puutteellinen, ja vaatii merkittäviä muutoksia.

Kun yritys tekee strategisen täyskäännöksen, se aloittaa prosessin alusta. Pivot on muutos, joka suunnitelmallisesti testaa uutta hypoteesia tuotteesta tai liiketoimintamallista. Täyskäännöksiä on useanlaisia: *Tarkennustäyskäännöksessä* (Zoom-in pivot) strategia muutetaan siten, että yksittäisestä ominaisuudesta tulee kokonainen tuote. *Loitonnustäyskäännöksessä* (Zoom-out pivot) on vastakkainen tilanne. Tämän tyyppisessä täyskäännöksessä kokonaisesta tuotteesta tulee laajemman tuotteen yksi ominaisuus. *Asiakassegmenttitäyskäännöksessä* (Customer segment pivot) yritys vaihtaa asiakassegmenttiä. Tällaiseen tilanteeseen voidaan joutua, jos tuotteen todetaan toimivan, mutta ennalta ennakoidusta poikkeavalla asiakaskunnalla. *Asiakastarvetäyskäännöksellä* (Customer need pivot) tarkoitetaan asiakkaiden tarpeiden muutoksen havaitsemista silloin, kun yrityksen tuotteen käyttäjäkunnalta löydetään uusia ongelmia, jotka ovat tärkeämpiä kuin mitä varten tuote oli alunperin suunniteltu. *Alustan täyskäännöksessä* (Platform pivot) tuote muutetaan alustaksi tai alusta tuotteeksi. *Liiketoiminta-arkkitehtuurin muutoksessa* (Business architecture pivot) yritys muuttaa liiketoiminta arkkitehtuurinsa. Yritys voi esimerkiksi muuttaa strategiansa yritykseltä kuluttajalle -mallista (B2C) yritykseltä yritykselle -malliin (B2B). *Kasvustrategian täyskäännöksessä* (Engine of growth pivot) yritys muuttaa kasvustrategiansa esimerkiksi nopeampaan kasvuun. *Kanavamuutoksessa* (Channel pivot) yritys pyrkii myymään samaa tuotetta tai palvelua eri jakelukanavan kautta.

Teknologiatäyskäännöksessä (Technology pivot) yritys muuttaa käyttämäänsä teknologiaa saadakseen samat tulokset tehokkaammin käyttäen eri teknologiaa.

Muutoksen tekeminen on yksi Lean Startup -menetelmän ydintoiminnoista. Onnistuneen muutoksen merkki on se, kun uuden strategian toteuttaminen tuottaa vahvistetun oppimisen tuloksia.

3 Tapaus AdPearl Oy

Pro Gradu tutkielman tavoitteena on tutkia, kuinka ketterät menetelmät sopivat yhteen startup-yritysten innovaatioprosssien kanssa. Tätä tutkielmaa peilataan ensimmäistä tuotettaan kehittävään yritykseen, AdPearl Oy:hyn. AdPearl Oy on startup-ohjelmistoyritys jonka päätuote on AdPearl.com-verkkosivusto. AdPearl.com palvelun visio on olla kattava tarjouspalvelu, joka kerää kuluttajille kauppojen tarjoukset keskitetysti yhteen paikkaan. Yrityksen ensimmäinen strategia vision saavuttamiseksi on koota kauppojen tarjoukset yhdelle verkkosivustolle. Verkkosivuston sisältö koostuu ainoastaan tarjouksista, jolloin kuluttaja tietää saavansa tuotteen hyvään hintaan. Yrityksen tavoite on kasvaa maailmanlaajuiseksi, useita maita kattavaksi tarjouspalveluksi.

Tarjouspalvelun alkuperäisen strategian mukaiset käyttäjäryhmät koostuvat kauppiaista sekä tarjouksista kiinnostuneista käyttäjistä. Kauppiaat syöttävät palveluun tarjouksensa, joita tarjouksista kiinnostuneet käyttäjät kuluttavat. Tarjouksista kiinnostuneet käyttäjät ovat passiivisia kuluttajia, jotka selaavat palvelua löytäen haluamansa tuotteet tarjoushintaan.

3.1 Liiketoimintamalli

AdPearl Oy:n visio on olla maailmanlaajuinen tarjouspalvelu, josta asiakkaat löytävät haluamansa tuotteet tai palvelut parhaaseen hintaan. Vision saavuttaminen vaatii tarkkaan määritellyn strategian. Yrityksen strategia kuvataan *liiketoimintamallilla* (business model) [OsP10]. Strategian toteuttamisen tuloksena syntyy tuote. Kuvasta 9 huomataan, että liiketoimintamallin tarkoituksena ei ole luoda kerralla valmista tuotetta, vaan tuote kehittyy liiketoimintamallin optimointiprosessien seurauksena. Optimointiprosessilla tarkoitetaan tuotteen muuttamista sen perusteella, millaisia tuloksia edellisen strategian toteuttaminen on tuottanut. Strategian muutos voi tulla ajankohtaiseksi jossain vaiheessa tuotekehitystä, mikäli vision saavuttaminen sen vaatii. Vaikka yrityksen tulisikin tehdä muutoksia, sen visio ei yleensä muutu. AdPearl Oy:n tapauksessa visio, olla maailman laajuinen tarjouspalvelu, ei siis muutu, vaikka strategia, eli nykyinen liiketoimintamalli ei kykenisi saavuttamaan tavoitetta. Tällöin on etsittävä uusia strategioita vision saavuttamiseksi.

Kuva 9: Visio, strategia ja tuote [Rie11].

AdPearl Oy:n liiketoimintamallin ydin on antaa asiakkaille (kauppiaille) palvelun *perustoiminnallisuus* (basic services) ilmaiseksi ja *lisätoiminnallisuus* (premium services) maksullisena. Tällaista liiketoimintamallia kutsutaan yleisesti *Freemiumiksi* [OsP10]. Freemium on yleinen liiketoimintamalli etenkin web-ohjelmistoja kehittävien yritysten keskuudessa.

Kauppiaat voivat parantaa tarjoustensa näkyvyyttä syöttämällä tarjouksensa palveluun ilmaiseksi. Kauppiaat voivat maksua vastaan ottaa käyttöön lisätoiminnallisuuksia, jotka parantavat tarjousten näkyvyyttä palvelussa. Kuluttajille palvelu on täysin ilmainen. Freemium-liiketoimintamallille on tyypillistä, että valtaosa asiakkaista käyttää palvelua ilmaiseksi ja vain pieni osa – yleensä alle 10 prosenttia – koko käyttäjäkunnasta käyttää maksullisia lisätoiminnallisuuksia [OsP10]. Lisätoiminnallisuuksia käyttävät asiakkaat rahoittavat koko palvelun. Tällainen liiketoimintamalli vaatii, että ilmaiseksi palvelua ilmaiseksi käyttävien asiakkaiden kustannukset ovat pienet.

Freemiumin kaksi olennaisinta mittaria ovat [OsP10]:

1. palvelua ilmaiseksi käyttävän asiakkaan keskimääräinen kustannus
2. ketkä ilmaiseksi käyttävistä asiakkaista siirtyvät käyttämään maksullisia lisäpalveluita

Osterwalder ja Pigneurr esittävät, että liiketoimintamalli on mahdollista esittää yhdeksän osa-alueen avulla [OsP10]. Osa-alueet kuvaavat kuinka yritys toimii ollakseen kannattava ja kattavat neljä liiketoimintamallin pääaluetta: *asiakkaat* (customers), *tarjooman* (offer), *infrastruktuurin* (infrastructure) ja *taloudellinen toteuttamiskelpoisuuden* (financial viability). Liiketoimintasuunnitelma koostuu *asiakassegmenttien* (customer segments), *arvolupausten* (value propositions), *kanavien* (channels), *asiakassuhteiden* (customer relationships), *tulovirtojen* (revenue streams), *avain resurssien* (key resources), *avaintoimintojen* (key activities), *avain suhteiden* (key partnerships) ja *kulurakenteen* (cost structure) kuvaamisesta [OsP10].

Kuva 10 esittää AdPearl Oy:n liiketoimintamallin graafisessa muodossa Osterwalderin ja Pigneurrin esittämän *liiketoimintamallipohjan* (business model canvas) avulla.

Asiakassegmentillä tarkoitetaan sitä asiakkaiden joukkoa, joita yritys palvelee. AdPearl Oy:n palvelu on *monipuolinen alusta* (multi-sided platform): asiakassegmenttejä ovat sekä passiiviset kuluttajat sekä kauppiaat. Passiiviset kuluttajat käyttävät palvelua kuluttaen kauppiaiden tuottamaa materiaalia (tarjouksia). Asiakassegmenttinä kauppiaat jakautuvat kahtia: maksaviin ja ilmaiseksi palvelua käyttäviin kauppiaisiin. Maksavat kauppiaat saavat tarjouksilleen lisänäkyvyyttä maksua vastaan. Jotta liiketoimintamalli toimii, tarvitaan sekä tarjouksista kiinnostuneita käyttäjiä (passiivisia kuluttajia) että kauppiaita.

Arvolupauksella tarkoitetaan sitoumusta, jonka avulla asiakkaan ongelma ratkaistaan ja tarve tyydytetään. AdPearl Oy:n palvelun arvolupauksena voidaan pitää *uutuutta* (newness), sillä vastaavaa palvelua ei vielä tämän muotoisena ole olemassa. Lisäksi palvelun käyttäjät ovat lähtökohtaisesti saapuneet etsimään tarjoushintaisia tuotteita tai palveluja, joka on mainostajalle mieluisa lähtökohta. Ilmaisuus on myös houkutin liittyä palvelun käyttäjäksi. Lisätoiminnallisuuksista maksaville kauppiaille luvataan parempaa näkyvyyttä palvelun sisällä, esimerkiksi hakutuloksissa.

Arvolupaukset toimitetaan asiakkaille kommunikaatio-, jakelu- ja myyntikanavien kautta. Kunkin kanavan pitää täyttää osa tai kaikki seuraavista vaiheista:

1. Kuinka yrityksen tuotteen tai palvelun näkyvyyttä saadaan parannettua,
2. kuinka auttaa asiakasta arvioimaan yrityksen arvolupauksia,
3. kuinka mahdollistaa asiakas ostamaan tiettyjä tuotteita tai palveluita,
4. kuinka arvolupaus välitetään asiakkaalle ja
5. kuinka oston jälkeistä asiakasta tuetaan

AdPearl Oy:n kommunikaatiokanavia ovat AdPearl.com-verkkosivusto, blogit sekä sosiaalisen median palvelut, kuten Facebook, Twitter ja LinkedIn. Jakelukanavana toimii AdPearl-verkkosivusto, jonka kautta kauppiaat pystyvät hankkimaan lisäpalveluja. Lisäksi kauppiaita lähestytään kohdistetulla sähköpostimarkkinoinnilla.

Jokaisen asiakassegmentin asiakassuhde solmitaan ja suhdetta pidetään yllä. Olennaista on määrittää jokaiselle asiakassuhteelle vaatimukset ja odotukset – mitä yritys haluaa asiakassuhteelta ja mitä asiakas odottaa yritykseltä. Kauppiaita ja kuluttajia houkutellaan palvelun pariin ja hankitut kauppiaat pyritään säilyttämään parhaan mukaan. Kauppiaita houkutellaan sitoutumaan palveluun mahdollistamalla yhteistyö palvelua kehitettäessä: kauppiaiden toiveita kuunnellaan ja palvelua rakennetaan toiveiden mukaisesti. Lisäksi AdPearl-palvelu sisältää työkaluja kuluttajien ja kauppiaiden väliseen kommunikaatioon. Kuluttajat voivat jättää kysymyksiä koskien tuotetta tai tuotealuetta, joihin vastaamalla kauppiaat voivat osoittaa ansiantuntemustaan kyseisellä tuotealueella.

Freemium-liiketoimintamallin kannalta on tärkeää laskea passiivisen kuluttajan kustannukset ja suhde, kuinka monta passiivista kuluttajaa yksi maksava asiakas kustantaa. Oletetaan, että 100 eurolla pystyy ylläpitämään 2000 passiivista kuluttajaa kuukaudessa. Yksi kuluttaja maksaa yritykselle 0.05 euroa kuukaudessa. Tällöin 1 maksava asiakas kustantaa 2000 ilmaista asiakasta, jos se maksaa yli 100 euroa kuukaudessa palvelun lisätoiminnallisuuksista. Asiakaskohtaiset kustannukset on mahdollista pitää pienenä, kun kaikki lähes kaikki palvelut on automatisoitu. Automatisoiminen on tärkeää palvelun kasvupotentiaalin kannalta.

Asiakkaalle onnistuneesti tarjotusta arvolupauksesta syntyy tulovirtaa. AdPearl Oy:n tulovirta rakentuu premium-ominaisuuksien *käyttömaksuista* (usage fees). Mitä enemmän asiakas käyttää lisätoiminnallisuuksia, sitä enemmän asiakas maksaa. Tällaisia ominaisuuksia ovat esimerkiksi tarjousten nostaminen palvelun parhaille paikoille. Tulovirtaa kartutetaan myös kolmannen osapuolen mainostajille kohdistetulla

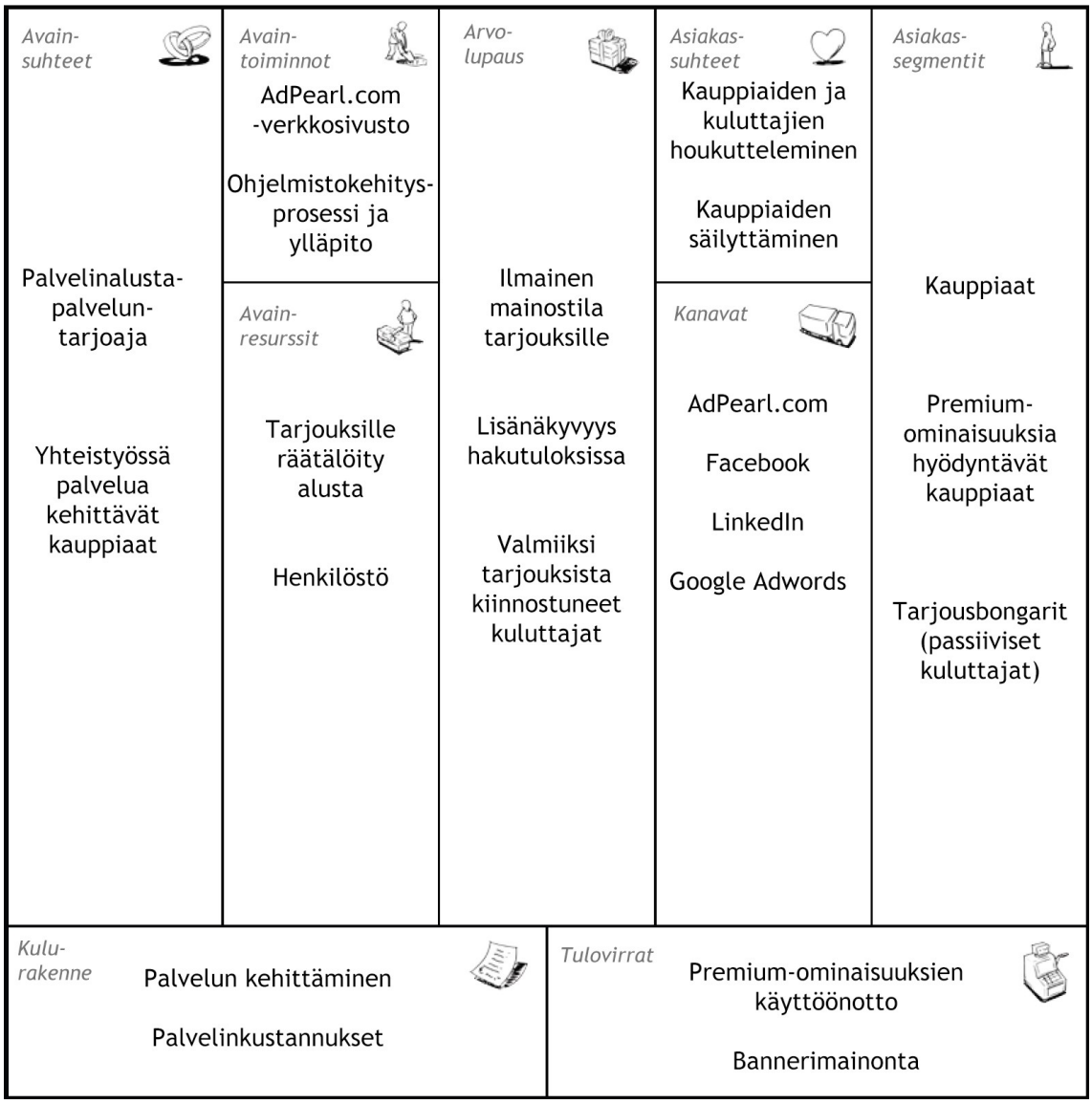
bannerimainosmyynnillä. Bannerimainonta tarkoittaa verkkosivujen sisälle upotettujen kolmannen osapuolen staattisia kuvia tai animoituja esityksiä. Banneri ohjaa käyttäjän mainostajan valitsemalle verkkosivulle.

Avainresurssi on sellainen kilpailuvaltti, jonka avulla asetetut tavoitteet pystytään saavuttamaan. AdPearl Oy:n avainresursseja ovat uudenlainen tarjouksille räätälöity alusta ja henkilöstö. Pienenä toimijana palautteeseen on mahdollista reagoida nopeasti. Henkilöstön ikä ja koulutustaso, joka mahdollistaa nykyaikaisten markkinointikanavien sekä teknologioiden hyödyntämisen.

Avaintoiminnot ovat sellaisia toimintatapoja, jotka mahdollistavat tavoitteiden saavuttamisen. AdPearl Oy:n tapauksessa AdPearl.com-verkkosivuston lisäksi yksi merkittävimmistä avaintoiminnoista on ohjelmistokehitysprosessi, johon keskitytään tarkemmin kappaleissa 4 ja 5.

Avainsuhteilla tarkoitetaan niitä toimintoja ja resursseja, jotka yritys on ulkoistanut toisten hoidettavaksi. Yrityksen kannalta ei ole järkevää omistaa itse kaikki käyttämiään resursseja [OsP10]. AdPearl Oy:n tapauksessa erään avainresurssin – palvelinalustan – ulkoistaminen on palvelun alkuvaiheessa halvempaa, kuin oman palvelimen ylläpitäminen. Web-sovelluksia valmistaville yrityksille yksi tärkeimmistä avainsuhteista on suhde palveluntarjoajaan. Sovelluksen on oltava käytettävissä vuorokauden ympäri ja palvelinalustan on oltava vikasietokykyinen.

Kulurakenne koostuu liiketoimintamallin osatekijöistä. AdPearl Oy:n kulurakenne pyritty minimoimaan. Yrityksen päämarkkinointikanava on sosiaalinen media. Rahaa vaativa markkinointi kohdistetaan tarkkoja tilastoja tuottaviin palveluihin kuten Google Adwordsiin ja Facebook-mainontaan. Markkinoinnin toimivuus pyritään mittaamaan tarkasti, ennen suuria mainoskampanjoita. Mittaamiseen käytetään Lean Startup -menetelmän luo-mittaa-opi -prosessia. Avainresursseista markkinoinnin lisäksi palvelinalusta sekä tarvittavat ohjelmistolisenssit tuottavat eniten kustannuksia.



Kuva 10: AdPearl Oy:n liiketoimintamalli Osterwalderin ja Pigneurrin esittelemän liiketoimintamallikankaalla kuvattuna [OsP10].

3.2 AdPearl Oy:n vaatimukset ohjelmistokehitysprosessille

AdPearl Oy:n tuote on verkkosivusto, josta kuluttajat löytävät kauppiaiden syöttämiä tarjouksia. Tuotteen kehityksen tukena käytetään ohjelmistokehitysprosessia. Ohjelmistokehitysprosessin tarkoitus on parantaa tuotteen laatua ja hyödyntää ihmisten kommunikointia ja ymmärrystä, tukea prosessin kehittymistä ja hallintaa [IEE04].

Ohjelmistokehitysprosessi on yksi AdPearl Oy:n liiketoimintamallin tärkeimmistä avaintoiminnosta. Käsiteltävät prosessit rajataan ketteriin menetelmiin, sillä ne

kykenevät reagoimaan nopeasti muutoksiin. Olemassa olevia ketteriä ohjelmistokehitysprosesseja arvioidaan AdPearl Oy:lle olennaisten innovaatiomenetelmien ja liiketoimintamallin kannalta merkittävien vaatimusten avulla luvussa 4. Käsiteltävät ketterät ohjelmistokehitysprosessit ovat XP, Scrum, Kanban ja FDD. Kehitysprosessin vaatimukset on kirjattu taulukkoon 1 priorisoituna AdPearl Oy:n kannalta tärkeimmästä vaatimuksesta vähiten tärkeään. Taulukon vaatimukset on jaettu innovoinnin ja liiketoimintamallin kannalta olennaisiin vaatimuksiin. Taulukon 1 menetelmät perustuvat luvussa 2 esiteltyihin menetelmiin.

3.2.1 Liiketoimintamallin vaatimukset

Yrityksen liiketoimintamalli voi vaatia muutoksia ajan kuluessa. Yrityksen strategia voi muutta merkittävästi, jonka avulla yrityksen visio pyritään saavuttamaan. Ohjelmistoprosessin on oltava kykenevä mukautumaan isoihin muutoksiin kesken tuotteen kehityksen. AdPearl Oy:n ensimmäinen liiketoimintamallista ammentava vaatimus on ketteryys. Tässä tutkielmassa keskitytään ainoastaan ketteriin ohjelmistokehitysprosesseihin. Ketterät ohjelmistokehitysprosessit pyrkivät olemaan kevyitä, tehokkaita, asiakaslähtöisiä ja nimensä mukaisesti ketteriä sopeutumaan muutoksiin [ASR02]. Prosessin on kyettävä tukemaan liiketoimintamallin optimointia ja muutoksia.

Passiivisen tuottaja-kuluttaja -suhteen sijaan prosessissa halutaan hyödyntää kuluttajien osallistumista tuotekehityksen aikana. Kaikkien kuluttajien kokemukset ovat tärkeitä oikeanlaisen tuotteen saavuttamiseksi. On olennaista, että käyttäjät kokevat tuotteen tai palvelun itselleen hyödyllisenä ja käytettävänä. Tästä johtuen sidosryhmien huomioimiseen halutaan kiinnittää erityistä huomiota. Sidosryhmien palautteeseen ei kuitenkaan luoteta sokeasti, vaan kaikki uudet innovaatiot halutaan todentaa toimiviksi vahvistetun oppimisen avulla. Vahvistettu oppiminen perustuu Lean Startup -menetelmän luo-mittaa-opi -prosessiin, joka on AdPearl Oy:n tärkein yksittäinen vaatimus, jonka ohjelmistokehitysprosessin on täytettävä.

Turhuuden karsiminen ohjelmistokehitysprosessista on olennaista. Prosessista on poistettava kaikki, mikä ei lisää arvoa asiakkaalle. Lisäksi prosessimallin on kyettävä skaalautumaan, eli vastaamaan yrityksen, tuotteen sekä kehitystiimin kasvuvaatimuksiin.

3.2.2 Innovaatiovaatimukset

AdPearl Oy pyrkii toiminnassaan innovatiivisuuteen, eli kykyyn kääntää luovuus todelliseksi tuotteeksi, palveluksi tai käytännöksi. Yrityksen vision saavuttaminen vaatii unelmointia ja ideointia, jota on myös ohjelmistoprosessin osalta tuettava. Suunnitteluajattelulle tyypillinen prototyyppien jatkuva tuottaminen on olennainen osa luo-mittaa-opi -prosessia. Suunnitteluajattelu pyrkii tuntemaan läpikotaisesti palvelun tai tuotteen käyttäjän ja sen toimintaympäristön. Lean Startup -menetelmän minimaalisten toimivien tuotteiden kehittämisen voidaan nähdä eräänlaisena prototyyppien tuottamiskeinona.

Käytettävyys on noussut yhä tärkeämmäksi osa-alueeksi ohjelmistojen laatua mitattaessa. Käytettävyyden tueksi on noussut käsite *käyttäjäkeskeinen kehitys* (User Centered Development). Käyttäjäkeskeisen lähestymistavan ja yleisen ohjelmistokehitysprosessin yhdistäminen on todettu hankalaksi [Mol06]. Käyttäjäkokemuksen huomioiminen on AdPearl Oy:lle tärkeä vaatimus.

AdPearl Oy kokee myös tärkeänä, että palvelu tuotteistetaan, eli palvelu viestii kokemuksesta ja palvelu on selkeä kokonaisuus, josta on helposti nähtävistä mitä se sisältää. Palvelun tuotteistaminen on osa palvelumuotoilua, joka näkee tärkeänä kaikkien sidosryhmien huomioimisen palvelun kehittämisessä. Täydellisellä palvelukokemuksella tarkoitetaan sitä, että palvelu nähdään kokonaisvaltaisena kokemuksena, joka voi alkaa jo kauan ennen kuin asiakas on yhteydessä palveluntarjoajaan ja jatkuu vielä palvelutilanteen päättymisen jälkeen.

AdPearl Oy on monipuolinen alusta, eli se yhdistää kaksi asiakasryhmää yhteen palveluun [Osp10]. AdPearl Oy:n kannalta tällainen liiketoimintamalli vaatii toteutuakseen sekä kauppiaita että kuluttajia. Passiiviset kuluttajat viettävät palvelussa aikaa kuluttaen kauppiaiden tuottamaa materiaalia, eli tarjouksia. Tulevaisuudessa alusta voisi myös tukea avointa innovointia. Prosessin olisi hyvä tukea omien ideoiden jakamista muille ja muiden ideoiden hyödyntämistä osana omaa liiketoimintamallia.

Viimeisenä vaatimuksena ohjelmistokehitysprosessille on huomioida ja kartoittaa yrityksen sisäinen harrastelijatieto kehitettävästä tuote- tai palvelualueesta. Henkilökunnan sisäiset vahvuudet on löydettävä.

Vaatimus	Prioriteetti	Menetelmä
Liiketoimintamalli		
Luo-mittaa-opi -prosessi	1	Lean Startup
Ketteryys	2	Ketterät menetelmät
Turhuuden karsiminen	5	Lean
Skaalautuvuus	6	Startup
Innovointi		
Unelmointi, ideointi	3	Millerin innovaatioprosessi
Prototyyppien jatkuva tuottaminen	4	Suunnitteluajattelu, Lean Startup (MVP)
Käyttäjäkokemuksen huomioiminen	7	Käyttäjäkeskeinen kehitys, Palvelumuotoilu
Palvelun tuotteistaminen	8	Palvelumuotoilu
Täydellinen palvelukokemus	9	Palvelumuotoilu
Alusta uusille, kolmannen osapuolen innovaatioille	10	Avoin innovointi
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	11	Harrastelijatieto

Taulukko 1: AdPearl Oy:n vaatimukset ohjelmistokehitysprosessille.

4 Ketterien menetelmien soveltuvuus AdPearl Oy:n vaatimuksiin

Tässä luvussa tutustutaan siihen, kuinka luvussa 2 esitellyt ketterät menetelmät täyttävät AdPearl Oy:n luvussa 3 asetetut vaatimukset.

4.1 XP:n soveltuvuus

Extreme Programming-menetelmä soveltuu hyvin startup-yrityksen tarpeisiin ja vaatimuksiin, mikäli projektilla on selkeä asiakas. Startup-yritykselle tällainen tilanne on riskialtis [Rie11]. Useasti startup-yritys ei vielä edes tiedä, kuka asiakas on.

Prosessimallin on kuitenkin mukautuva AdPearl Oy:n kaltaiseen tilanteeseen, jossa asiakas on yritys itse ja tuotteen käyttäjäkunta on heterogeeninen tai vielä osittain tuntematon. XP-ohjelmistokehityksen ensimmäinen julkaisu, perustarpeiden tyydyttämiseen pohjautuva iteraatio voidaan nähdä Lean Startup -menetelmän MVP:n toteuttamisena. XP näkee, että iteraatioiden tarkoitus on kehittää tuote yritys- ja oppimisprosessien seurauksena. Tällä ei kuitenkaan tarkoita sellaista oppimista, mitä Lean Starup-menetelmä painottaa. Oppimiskokemukset ovat yrityksen sisäisiä, tai tuotteen asiakkaalta saatuja subjektiivisiä kokemuksia, ei empiiriseen dataan pohjautuvia objektiivisia havaintoja. Ensimmäisen julkaisun – pienimmän julkaisukelpoisen tuotteen toteuttamisen – tuloksesta voidaan kuitenkin kerätä palautetta tuotteen käyttäjäkunnalta, joko ei-insinöörillisiä taitoja käyttäen suoralla yhteydenpidolla tai mittaamalla käyttäjäkunnan käyttäytymistä tuotteen sisällä erilaisin mittarein. Tällaisia mittareita voivat olla esimerkiksi palveluun rekisteröityneiden käyttäjien määrä suhteessa koko kävijämäärään tai aktiivisten käyttäjien määrä suhteessa koko rekisteröityneeseen käyttäjäkuntaan. Yhteydenpidon sekä kertovat omalta osaltaan myös käyttäjäkokemuksesta. Tällöin prosessi saadaan tukemaan Lean Startup -menetelmää.

Samoja mittareita voidaan käyttää hyväksi seuraavien iteraatioiden tuloksen mittaamiseen. Tulosten perusteella opitaan, mikä tuotteessa toimii ja mikä kaipaa muutosta. XP on lähtökohtaisesti inkrementaalinen prosessimalli, eli lopullinen tuote kasvaa iteraatio iteraatiolta, lisäten olemassa olevaan tuotteeseen ominaisuuksia. Inkrementaalisuus ei välttämättä ole ensimmäistä tuotettaan kehittävälle yritykselle aina mahdollinen toimintatapa: joskus tuotteen ominaisuuksia on syytä muuttaa radikaalisti,

jolloin aiemmin toteutetun ohjelmakoodin hylkääminen voi olla ajankohtaista. XP ei tähän kannusta, muttei aseta siihen myöskään estettä.

XP on rakennettu tukemaan innovaatioita tuotekehityksessä [GSW06]. AdPearl Oy:n innovaatiovaatimuksista toteutuu Millerin innovaatioprosessin unelmointi ja ideointi-vaihe. Unelmointi ja ideointi voidaan ottaa huomioon, kun käyttäjäkertomuksia laaditaan seuraavaa julkaisua varten. Nopeat julkaisut voidaan toteuttaa nopeiden prototyyppien omaisesti, toteuttaen vain olennaisimmat ominaisuudet. Myös käyttäjäkokemus on mahdollista huomioida jo suunnitteluvaiheessa ja sen toimivuutta voidaan mitata aiemmin mainittujen mittareiden avulla.

Innovoinnin kannalta kaikkien XP:n toimintatapojen noudattaminen voi olla vaikeaa [GSW06]. Esimerkiksi asiakkaan täydellisen läsnäolon saavuttaminen voi olla ongelmallista: asiakas ei välttämättä ole tulevan tuotteen loppukäyttäjä. Tällaisessa tapauksessa käyttäjäkertomukset pitäisi muodostaa yhdessä loppukäyttäjän, ei tuotteen tilaajan kanssa. Myöskään useat asiakasyritykset eivät ole valmiita luovuttamaan yhtä työntekijäänsä ohjelmistoyrityksen tiloihin. Lisäksi jatkuva yksikkötestien tuottaminen voi koitua ongelmalliseksi startup-yrityksen kannalta, sillä yksikkötestien jatkuva tuottaminen vie enemmän aikaa, kuin pelkän järjestelmätestauksen toteuttaminen. Esimerkiksi MVP:tä kehitettäessä, kaiken kattavaan yksikkötestaamiseen ei ole perusteita. Sama pätee pariohjelmointiin. Pariohjelmointi parantaa ohjelmiston laatua, mutta vaatii kaksi tekijää. Startup-yrityksen henkilöstön määrä voi muodostaa esteen pariohjelmoinnin toteuttamiseksi.

Pienet julkaisut, jatkuva integrointi, koodin yhteisomistajuus ja refaktorointi puolestaan soveltuvat hyvin startup-yritykselle. Startup-yrityksille nopea oppiminen ja omien toimintatapojen vahvistaminen on olennaisen tärkeää. Pienet julkaisut toimivat tähän tarkoitukseen hyvin ja sopivat luonteeltaan luo-mittaa-opi -menetelmää toteuttavaksi tuotantoonvientitavaksi. Jatkuva integrointi ja koodin refaktorointi taas ylläpitävät ohjelmiston laatua, joka pienintä toimivaa tuotetta kehittäessä ja nopeita prototyyppejä luodessa on voinut kärsiä.

Startup-yritykselle on tyypillistä tähdätä kasvuun, jolloin ohjelmistokehitysprosessin on kyettävä skaalautumaan samanaikaisesti. XP on suunnattu pienille ja keskikokoisille kehitystiimeille, joiden jäsenmäärä koostuu optimitilanteessa 3-20 henkilöstä. XP toimii parhaiten silloin, kun kommunikaatio tiimin jäsenten välillä on kokoajan mahdollista. Tästä johtuen tiimin on tärkeää sijaita fyysisesti lähellä toisiaan [ASR02]. Tiimi

hajottaminen kahteen eri kerrokseen tai jopa eri paikkoihin yhden kerroksen sisällä, on epäsuotuisaa [ASR02]. Tämä asettaa omat rajoituksensa skaalautuvuudelle: kun yrityksen henkilöstö kasvaa ja projekti laajenee, voi läheinen sijainti ja välitön kommunikaatio tuottaa hankaluuksia. Toisaalta yhteiset koodikäytännöt tukevat skaalautuvuutta: kun startup-yritykset lähtökohtaisesti haluavat kasvaa ja kansainvälistyä - on ohjelmakoodin ja muun dokumentaation oltava alusta alkaen yhtenäistä ja selkeää.

Mikäli XP:n toimintatapoja noudatetaan järjestelmällisesti, se ei ole omiaan Lean ajattelun mukaisen turhuuden karsimisen kannalta. XP:n ehdoton hyöty on se, että sen avulla toteutettu ohjelmisto ei välttämättä ole sitä mitä projektin alussa kuviteltiin, mutta se täyttää asiakkaan vaatimukset [ASR02]. Taulukkoon 2 on kuvattu AdPearl Oy:n ohjelmistokehitysprosessille asetetut vaatimukset XP-menetelmän osalta.

Extreme Programming	
Vaatimus	**Täyttyykö vaatimus (Kyllä / Osittain / Ei)**
Liiketoimintamalli	
Luo-mittaa-opi -prosessi	**Kyllä**
Ketteryys	**Kyllä**
Turhuuden karsiminen	**Osittain**
Skaalautuvuus	**Osittain**
Innovointi	
Unelmointi, ideointi	**Kyllä**
Prototyyppien jatkuva tuottaminen	**Kyllä**
Käyttäjäkokemuksen huomioiminen	**Osittain**
Palvelun tuotteistaminen	**Ei**
Täydellinen palvelukokemus	**Ei**
Alusta uusille, kolmannen osapuolen innovaatioille	**Ei**
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	**Kyllä**

Taulukko 2: AdPearl Oy:n vaatimusten [taulukko 1] toteutuminen XP-menetelmän avulla.

4.2 Scrumin soveltuvuuus

Scrum on kehikko, joka räätälöidään silloiselle projektille sopivaksi. Joustavuutensa takia Scrum on AdPearl Oy:lle sopivampi kuin XP.

AdPearl Oy:n tärkein yksittäinen vaatimus, vahvistettu oppiminen, vaatii, että tuotteesta tehdään MVP niin nopeasti kuin mahdollista. MVP:tä testataan todellisessa toimintaympäristössään todellisilla asiakkailla, jolloin turhilta oppimiskokemuksilta vältytään. Vahvistettu oppiminen taas vaatii luo-mittaa-opi -prosessin iteroimista. Scrumin sprintteihin perustuva rakenne toimii hyvin tällaisessa tilanteessa. Scrumin lähtökohta on se, että tuote rakennetaan inkrementaalisesti, eli tuotteen ominaisuudet täydentyvät iteraatio iteraatiolta [ASR02]. Scrum kannustaa laatuun jokaisen julkaisun osalta. MVP:n ei lähtökohtaisesti tarvitse olla laadultaan täydellinen, vaan minimaalinen toimiva tuote, omalla tavallaan prototyyppi tulevasta tuotteesta tai palvelusta.

Sprintin edistymiskäyrä mittaa ainoastaan työtehtävien toteutumista, mikä on hyvä asia työtehtävien toteutumisen kannalta. *Valmiin määritelmä* sen sijaan on AdPearl Oy:n kannalta tärkeämpi mittari. Kehitystiimin ja tuoteomistajan yhdessä laatima valmiin määritelmä voidaan yhdistää Lean Startup -menetelmän vahvistetun oppimisen kanssa – tällöin ominaisuuden voidaan ajatella olevan valmis, vasta kun se tuottaa vahvistetun oppimisen tuloksia.

Suunnittelupelin sprinttikohtaiset suunnittelukokoukset mahdollistavat ideoinnin ja unelmoinnin. Kun järjestelmää määritellään, voi suunnittelun tukena käyttää Millerin innovaatioprosessin unelmointivaiheen kysymyksiä. Unelmoinnin tuloksena syntynyt konsensusunelma voidaan osittaa kehitysjonolistaan. Unelmoinnin lisäksi kehitysjonolistaan voidaan sisällyttää vaatimuksia tarpeen mukaan useilta eri sidosryhmiltä: asiakkailta, myynti- tai markkinointiosastolta, asiakastuelta tai ohjelmistokehittäjiltä. Kehitysjonolistan tehtävät voivat siis olla tilanteesta riippuen erilaisia: vaatimuksia, tarinoita tai ominaisuuksia.

Scrum-tiimissä tulisi olla alle 10 henkilöä [ScB02]. Kun yritys kasvaa, Scrum kykenee skaalautumaan**:** Scrumin prosessimalli mahdollistaa useampien tiimien yhteystyön samassa projektissa [ASR02]. Scrum-projektissa tuoteomista ohjaa kehitystä. Tuoteomistaja ohjaa projektin siihen suuntaan, jonka visioidensa ja mahdollisten asiakkaiden tarpeiden perusteella näkee parhaaksi [ASR02]. Pienissä startup-yrityksissä, kuten AdPearl Oy:ssä, tuoteomistaja on usein yrittäjä itse. Tuoteomistajalla on visio,

joka pyritään saavuttamaan sprintin aikaisen strategian avulla. Tuoteomistajan rooli on kriittinen – hän käytännössä valitsee sprintteihin sisällytettävät ominaisuuden kehitysjonolistasta. Scrumin räätälöitävyys mahdollistaa myös taulukon 3 kaltaisten innovaatiovaatimusten nostamisen sprintin kehitysjonolistaan. Tarpeen mukaan äärimmäiset käyttäjiin, käyttäjäkokemukseen tai palvelun tuotteistamiseen voi kiinnittää erityishuomiota.

Scrumissa on kuitenkin muutamia puutteita, jotka heikentävät nopeaa reagointia äkillisiin muutoksiin. Aikasidonnaiset sprintit mahdollistavat kehitystiimin toteuttamaan ennalta sovitun määrän tehtäviä, tehtävien prioriteetteja vaihdellen. Scrum kuitenkin vastustaa muutoksia iteraatioiden aikana [Kni10]. On myös hyvin epätodennäköistä, ettei aloittelevan yrityksen ohjelmistokehitysprosessin vaatimukset muuttuisi lainkaan yhden kuukauden aikana, joka on tyypillinen sprintin pituus. Scrumissa ei ole mahdollista lisätä työtehtäviä kesken iteraation. Scrumtiimi tyypillisesti ei pysty lisäämään tehtävää E iteraation tehtävälistaan, sillä tiimi on jo sitoutunut toteuttamaan tehtävät A, B, C ja D kyseisen sprintin aikana [Kni10]. Tästä johtuen tehtävä E ainoastaan lisätään tuotteen kehitysjonolistaan. Jos tuoteomistaja kokee tehtävän tärkeäksi, tehtävä pystytään ottamaan tuotteen kehitysjonolistasta aikaisintaan seuraavan sprintin aikana toteutettavien tehtävien joukkoon.

AdPearl Oy:n yksi tärkeimpiä vaatimuksia ohjelmistokehitysprosessille on turhuuden minimointi. Scrum on prosessimalli, joka on sovellettavissa kulloiseen projektiin joustavammin kuin XP. Tämä mahdollistaa paremmin Lean ajattelun mukaisen turhuuden minimoinnin, kun projektinsisäiset toimintatavat on valittavissa joustavammin. Turhan työn tekeminen näkyy kehitystiimin tuottavuudessa ja yksi merkittävimmistä tuottavuuteen vaikuttavista tekijöistä on kehityksessä olevien työtehtävien määrä, sekä ylimääräisten ominaisuuksien kehittäminen [SBT09]. Työtehtävien rajoittaminen Scrum-prosessimallissa tulevan sprintin työtehtävät valitaan tuotteen kehitysjonolistasta. Työntekijöillä on taipumus ottaa enemmän tehtäviä sprintin kehitysjonolistaan, kuin mitä he todellisuudessa ehtivät tekemään [SBT09]. Työtehtävien määrän lieventäminen lisää tuottavuutta merkittävästi [SBT09]. Tästä johtuen Scrumiin olisi hyvä sisällyttää esimerkiksi *Kanban-taulun* [Hir08] tapainen työnkulun visualisoiva työkalu.

Kaiken kaikkiaan Scrum on AdPearl Oy:lle sopiva ohjelmistokehitysprosessi, keveytensä ja sovellettavuudensa ansiosta. Se myös tukee kaikkia AdPearl Oy:n

liiketoimintamallin vaatimuksia sekä on luonteeltaan innovatiivisuutta tukeva. Retrospektiivien ja sprinttien katselmointitapaamisten avulla projektin tila tarkistetaan säännöllisesti ja mahdollinen muutostarve tulee huomioitua useasti. Vaikka sprintin sisäisten muutosten tekeminen ei ole täysin joustavaa, mikään Scrum-prosessimallin konventio ei estä täyskäännöksen tekoa: uudet täyskäännöksen vaatimat ominaisuudet voidaan lisätä tuotteen kehitysjonolistaan, sekä ottaa seuraavan sprintin toteutettavien tehtävien joukkoon.

Scrum	
Vaatimus	**Täyttyykö vaatimus (Kyllä / Osittain / Ei)**
Liiketoimintamalli	
Luo-mittaa-opi -prosessi	**Kyllä**
Ketteryys	**Kyllä**
Turhuuden karsiminen	**Kyllä**
Skaalautuvuus	**Kyllä**
Innovointi	
Unelmointi, ideointi	**Kyllä**
Prototyyppien jatkuva tuottaminen	**Kyllä**
Käyttäjäkokemuksen huomioiminen	**Osittain**
Palvelun tuotteistaminen	**Osittain**
Täydellinen palvelukokemus	**Osittain**
Alusta uusille, kolmannen osapuolen innovaatioille	**Ei**
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	**Kyllä**

Taulukko 3: AdPearl Oy:n vaatimusten [taulukko 1] toteutuminen Scrum-prosessimallin avulla.

4.3 Kanbanin soveltuvuus

Kanban on joukko toimintaperiaatteita, eikä itsessään varsinainen prosessimalli. Kanban tarjoaa kuitenkin startup-yritykselle Lean ajatteluun sopivan kehyksen työtehtävien ohjaukseen.

Kanban sopii myös Lean Startup -menetelmän luo-mittaa-opi -prosessin työkaluksi [Rie11, Zhe12]. Luo-mittaa-opi -prosessille voidaan kuvata oma Kanban-taulunsa. Kuvassa 11 luo-mittaa-opi -prosessi on kuvattu Kanban-tauluun viidellä eri sarakkeella [Zhe12]:

1. Tuotannossa olevat ominaisuudet, joista ei ole vielä opittu.
2. Ominaisuudet, jotka eivät aiheuta haittaa nykyisen ohjelmiston toimivuudelle ja ohjelmiston tuottamalle liikevaihdolle.
3. Ominaisuudet, joiden parannusvaikutusta määritellään parhaillaan osalla asiakkaista. Kokevatko asiakkaat ratkaisun toimivaksi vai ei?
4. Ominaisuudet, jotka odottavat julkaisua koko asiakaskunnalle.
5. Vahvistetut ominaisuudet.

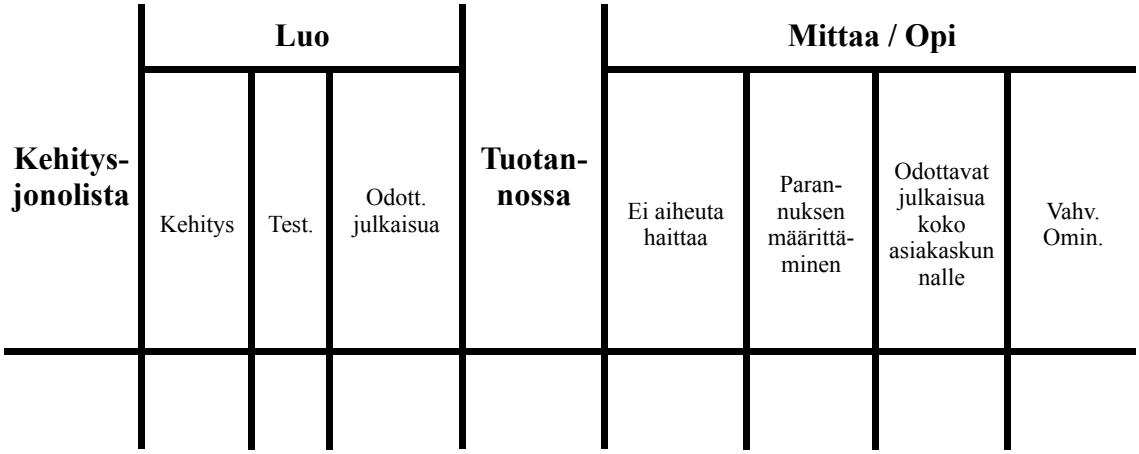
Kirjaimet A-L kuvaavat työtehtäviä.

Tuotannossa	Ei aiheuta haittaa	Parannuksen määrittäminen	Odottavat julkaisua koko asiakaskunnalle	Vahvistetut ominaisuudet
H *I* *J* *K* *L*	*F* *G*	*D* *E*	*C*	*A* *B*

Kuva 11: Luo-mittaa-opi -prosessin työnkulku Kanban-taululla kuvattuna [Zhe12].

Kuvasta 12 nähdään, kuinka ketterän Kanban-taulun ja Luo-mittaa-opi -prosessin taulun yhdistämällä aikaansaadaan yhdistetty Kanban-taulu, joka kattaa Lean Startup -menetelmän mukaisen kehitysprosessin kaikki vaiheet. Työnkulku etenee vasemmalta oikealle, kuten ketterässä Kanban-taulussa. Kaikki perinteisen ohjelmistotuotannon

mukaiset valmiit tehtävät, eli tehtävät jotka ovat tuotannossa, on vahvistettava. Vahvistamisella tarkoitetaan Lean Startup -menetelmän vahvistettua oppimista, joka perustuu oikeilta asiakkailta empiirisen datan keräämiseen. Dataa tarkastelemalla voidaan sanoa, onko kehitetty ominaisuus omiaan lisäämään asiakkaan kokemaa arvoa vai vähentämään sitä. Tehtävät, jotka odottavat julkaisua tai ovat jo tuotannossa on vahvistettava. Taulukon sarake ”ei aiheuta haittaa” kuvaa sellaisia ominaisuuksia, jotka eivät aiheuta haittaa asiakkaalle tai olemassa olevalle toiminnallisuudelle. Tehtävää, jota ei ole vahvistettu, ei voi poistaa Kanban-taululta. Vahvistaminen usein vaatii ei-insinöörillisiä taitoja, kuten tavoitteiden tarkastelua tai puhumista asiakkaille [Rie11].



Kuva 12: Yhdistetty luo-mittaa-opi Kanban-taulu [Zhe12].

Osa kehitetyistä ominaisuuksista voidaan aluksi julkaista ainoastaan osalle asiakaskunnasta, jolloin kehitetyn ominaisuuden vaikutusta voidaan peilata suhteessa sen hetkisen vallitsevan ominaisuuden tuloksiin. Mikäli uuden ominaisuuden tulokset ovat parempia kuin sen hetkisen ominaisuuden, ominaisuus voidaan vahvistetusti siirtää kuvan 12 Kanban-taulun viimeiseen sarakkeeseen.

Lean ja Kanban keskittyvät poistamaan viivettä eri työvaiheiden välillä. Tyypillisesti viive minimoidaan rajoittamalla kehityksessä olevien tehtävien määrää tai vaihtamalla järjestystä, jossa tehtävät suoritetaan. Viiveen minimoiminen näkyy Kanban-taululla, joka visualisoi projektin etenemisen. Kanban kykenee reagoimaan nopeasti muutoksiin, sillä uusien tehtävien lisääminen kehitysjonolistaan on mahdollista [Kni10]. Kehitysjonolista voi sisältää ainoastaan ennalta määrätyn määrän tehtäviä, joten uuden tehtävän lisäys usein vaatii jonkin tehtävän poistamisen listalta. Lean on parhaimmillaan parantamaan investoinnin tuottoprosenttia, joka on startup-yritykselle tärkeä mittari [Rie11]. Kanban on myös joustava yrityksen muuttuviin tarpeisiin. Uutta tuotetta

kehitettäessä Kanban voidaan toteuttaa iteratiivisesti ja inkrementaalisesti. Toisaalta esimerkiksi ylläpito ja tukitoimintojen toteuttamiseen iteratiivisuudelle ei välttämättä ole perusteita: kun ohjelmistoa integroidaan jatkuvasti tuotantoon, ei ensimmäisen ja ainoan ”iteraation” jälkeisille julkaisuille ole välttämättä perustetta [Kni10].

Kanban on luonteeltaan joustava. AdPearl Oy:n vaatimustaulukosta havaitaan, että useimmat vaatimukset täyttyvät. Vahvistettu oppiminen toteutuu ketterän ja luo-mittaa-opi Kanban-taulujen muodostaman prosessin avulla. Prosessi on soveltuva myös jatkuvien prototyyppien toteuttamiseen. Tuotantoprosessien parantamiseen perustuvan luonteensa johdosta Kanban ei sisällä menetelmiä innovoinnin, asiakkaiden tai palvelukokonaisuuksien tai erityishuomioimiseksi. Kanban itsessään ei tarjoa menetelmää vaatimusten keräämiselle tai innovoinnille, mutta ketterän Kanbanin iteratiivisuuden ansiosta menetelmään voidaan yhdistää myös Millerin innovaatioprosessin unelmointi ja ideointi-vaihe. Menetelmiä voidaan soveltaa esimerkiksi iteraation alussa tapahtuvan käyttäjäkertomusten keräämiseksi, ennen kuin kertomuksista ositetut tehtävät lisätään Kanban-taululle. Kanban ei myöskään ohjeista XP:n tai Scrumin tapaan jatkuvaan yhteydenpitoon asiakkaiden kanssa, mutta Kanban ei myöskään menetelmänä rajoita yhteydenpitoa. AdPearl Oy:n tilanteessa Kanban toimii parhaiten yhdistettynä jonkin toisen ohjelmistokehitysprosessin kanssa.

Kanban	
Vaatimus	**Täyttyykö vaatimus (Kyllä / Osittain / Ei)**
Liiketoimintamalli	
Luo-mittaa-opi -prosessi	**Kyllä**
Ketteryys	**Osittain**
Turhuuden karsiminen	**Kyllä**
Skaalautuvuus	**Ei**
Innovointi	
Unelmointi, ideointi	**Osittain**
Prototyyppien jatkuva tuottaminen	**Kyllä**
Käyttäjäkokemuksen huomioiminen	**Ei**
Palvelun tuotteistaminen	**Ei**
Täydellinen palvelukokemus	**Ei**
Alusta uusille, kolmannen osapuolen innovaatioille	**Ei**
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	**Osittain**

Taulukko 4: AdPearl Oy:n vaatimusten [taulukko 1] toteutuminen Kanban-menetelmän avulla.

4.4 Feature Driven Development-prosessimallin soveltuvuus

FDD sopii hyvin projektiin, jonka vaatimukset ovat tarkoin määritelty ja toimintaympäristö sekä asiakaskunta ovat entuudestaan selvillä [PaF02]. Prosessimalli painottaakin vaatimusten tarkkaa kartoittamista ja niiden ymmärtämistä prosessin alkuvaiheessa.

Startup-yrityksille on kuitenkin tyypillistä, että kehitettävän tuotteen toimintaympäristön käyttäytyminen on vielä osittain tuntematonta [Rie11]. FDD-prosessin ensimmäinen vaihe, yleismallin luominen, olettaa, että sovellusasiantuntijalla on tiedossa tulevan järjestelmän laajuus, konteksti, järjestelmävaatimukset sekä joukko dokumentoituja vaatimuksia. Lähtökohtaisesti Lean Startup -menetelmä näkee tällaiset oletukset mahdottomina, sillä menetelmä olettaa, että yrityksen visiota ei todennäköisesti tulla saavuttamaan ensimmäisellä strategialla. Tällöin prosessin nojaaminen ennalta määriteltyihin toiminta-alueen oletuksiin on vaarallista. FDD ei siis sovellu kovinkaan hyvin Lean Startup menetelmän luo-mittaa-opi -prosessin toteuttavaksi menetelmäksi.

FDD-prosessilla voi toteuttaa tuotteen ensimmäisen version, MVP:n, mutta MVP:stä opittujen muutosten tekeminen ei ole suoraviivaista. FDD-prosessi ei varsinaisesti ole kykenevä laajoihin muutoksiin tai täyskäännöksiin. Uusien ominaisuuksien lisääminen toimii samaan tapaan kuin Scrumissa: uudet havaitut ominaisuudet lisätään ominaisuusjoukkoon. Uudet ominaisuudet kaavoitetaan, eli ominaisuudesta laaditaan korkeantason suunnitelma. Kaavoittaminen vaatii, että tässä vaiheessa yleismallin luonnin yhteydessä ominaisuuksista on muodostettu luokat. Jos ominaisuudelle ei entuudestaan ole luokkaa, täytyy sellainen suunnitella. Luokat taas osoitetaan luokkaomistajille, jotka muodostavat ominaisuustiimin. Uuden ominaisuuden toteuttaminen siis vaatii, että uudesta ominaisuudesta vastaavat luokanomistajat ovat vapautuneet ominaisuustiimistä [ASR02]. Uuden ominaisuuden toteuttaminen vaatii siis vähintään yhden käynnissä olevan iteraation loppuun asti saattamisen.

Prosessi koostuu viidestä peräkkäisestä vaiheesta. FDD-prosessin iteratiivinen osuus mahdollistaa nopean reagoimisen viime hetken vaatimuksiin ja liiketoimintamallin muutoksiin. Iteratiivinen osuus ei kuitenkaan sisällä uusien ominaisuuksien kartoittamista, ainoastaan aiemmin valittujen ominaisuuksien suunnittelua ja toteuttamista. Tämä aiheuttaa ongelman: yleismallin, ominaisuuslistan ja ominaisuuksien kaavoittamisen laatimisen yhteydessä tuotteen tai palvelun

toimintaympäristö on osattava ottaa huomioon täydellisesti.

FDD antaa vapaat kädet ominaisuuksien ja vaatimusten kartoittamiseen, jolloin vaatimusten kartoittamiseen voi käyttää kyseiseen projektiin sopivaa menetelmää. AdPearl Oy:n tapauksessa yleismallin suunnittelussa voidaan käyttää apuna Millerin innovaatioprosessia. Yleismallin jälkeen luodaan ominaisuuslista, johon esitetään kaikki asiakkaalle arvoa tuottavat toiminnot. Kun ominaisuuslistaan kerätään vain asiakkaalle arvoa tuottavat ominaisuudet, noudattaa toimintamalli täydellisesti Lean ajattelun periaatteita.

Ominaisuustiimit työstävät kukin omia tehtäviään, jotka integroidaan erikseen osaksi laajempaa järjestelmää. Tämä mahdollistaa projektin skaalautumisen uusien lisäominaisuuksien suhteen. FDD:n ominaisuustiimit työstävät heille osoitettuja ominaisuuksiaan omalla iteraatiosyklillään, joka myös tukee skaalautuvuutta. Itsenäiset tiimit myös estävät pullonkaulojen muodostumisen, kun yhden ominaisuustiimin vaikeudet eivät suoraan heijastu muiden tiimien toimintaan.

FDD pitää sisällään toimintatapoja, joista osa on AdPearl Oy:n kannalta hyödyllisiä. Säännölliset tuotantoonviennit tukevat Lean Startup -menetelmän periaatteita. Myös tarkastukset voidaan soveltaa tukemaan Lean Startup -menetelmää. Tarkastukset voidaan suunnitella siten, että ne käsittelevät erilaisten, projektille luotujen mittareiden tarkastelemista tai yhteydenpitoa asiakkaisiin ja käyttäjiin. Ominaisuustiimit taas ovat oman ominaisuusalueensa asiantuntijoita, jolloin heillä on syvällinen tuntemus kehitettävästä aihealueesta sekä mahdollisesta käyttökontekstista. Tuntemus on kerätty kohdealueen oliomallintamis-toimitatavan avulla, jonka aikana pyritään tutkia, selittää ja mallintaa ominaisuuden kohdealuetta.

FDD on prosessimallina liian jäykkä AdPearl Oy:n asettamille vaatimuksille: luo-mittaa-opi -prosessia ei saavuteta täydellisesti.

Feature Driven Development	
Vaatimus	**Täyttyykö vaatimus (Kyllä / Osittain / Ei)**
Liiketoimintamalli	
Luo-mittaa-opi -prosessi	**Osittain**
Ketteryys	**Osittain**
Turhuuden karsiminen	**Kyllä**
Skaalautuvuus	**Kyllä**
Innovointi	
Unelmointi, ideointi	**Kyllä**
Prototyyppien jatkuva tuottaminen	**Osittain**
Käyttäjäkokemuksen huomioiminen	**Kyllä**
Palvelun tuotteistaminen	**Ei**
Täydellinen palvelukokemus	**Kyllä**
Alusta uusille, kolmannen osapuolen innovaatioille	**Ei**
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	**Kyllä**

Taulukko 5: AdPearl Oy:n vaatimusten [taulukko 1] toteutuminen FDD-prosessimallin avulla.

Taulukkoon 6 on koottu yhteen taulukkojen 2-5 sisällöt, eli taulukon 1 vaatimusten toteutuminen XP-, Scrum-, Kanban- ja FDD-menetelmien osalta.

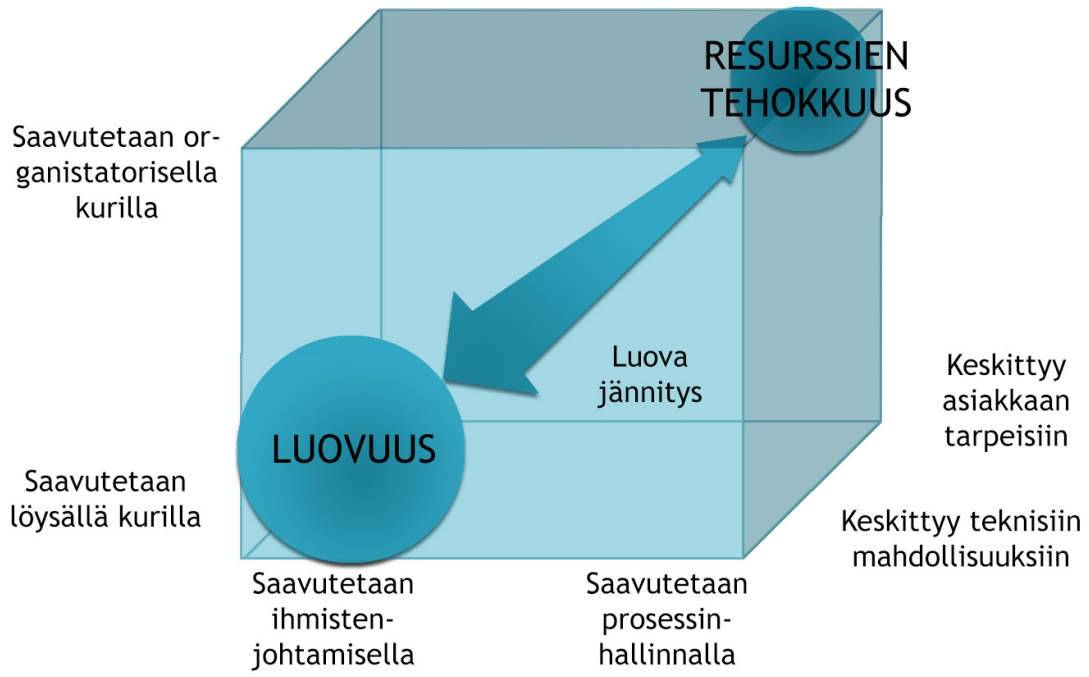
Yhteenveto luvun 4 ohjelmistokehitysprosesseista				
Vaatimus	**Täyttyykö vaatimus (Kyllä / Osittain / Ei)**			
	XP	Scrum	Kanban	FDD
Liiketoimintamalli				
Luo-mittaa-opi -prosessi	**Kyllä**	**Kyllä**	**Kyllä**	**Osittain**
Ketteryys	**Kyllä**	**Kyllä**	**Osittain**	**Osittain**
Turhuuden karsiminen	**Osittain**	**Kyllä**	**Kyllä**	**Kyllä**
Skaalautuvuus	**Osittain**	**Kyllä**	**Ei**	**Kyllä**
Innovointi				
Unelmointi, ideointi	**Kyllä**	**Kyllä**	**Osittain**	**Kyllä**
Prototyyppien jatkuva tuottaminen	**Kyllä**	**Kyllä**	**Kyllä**	**Osittain**
Käyttäjäkokemuksen huomioiminen	**Osittain**	**Osittain**	**Ei**	**Kyllä**
Palvelun tuotteistaminen	**Ei**	**Osittain**	**Ei**	**Ei**
Täydellinen palvelukokemus	**Ei**	**Osittain**	**Ei**	**Kyllä**
Alusta uusille, kolmannen osapuolen innovaatioille	**Ei**	**Ei**	**Ei**	**Ei**
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	**Kyllä**	**Kyllä**	**Osittain**	**Kyllä**

Taulukko 6: AdPearl Oy:n vaatimusten [taulukko 1] toteutuminen luvussa 4 esiteltyjen prosessimallien avulla.

5 AdPearl Oy:lle räätälöity ohjelmistokehitysprosessi

Liikeidean ja ohjelmistokehityksen suhde on kiinteä. Kehitetty liikeidea pitäisi saada käytäntöön – on siis olemassa joukko arvauksia, ongelmia, jotka pitää ratkaista. AdPearl Oy:n yksi tärkeimmistä avaintoiminnoista arvausten ratkaisemiseksi on luova ja tehokas ohjelmistokehitysprosessi.

Yrityksen avaintoiminnot ovat parhaimmillaan sekä luovia että tehokkaita. *Luovuuden* (Creativity) ka *tehokkuuden* (Efficiency) saavuttaminen on kuitenkin usein ristiriidassa keskenään [GSW06]. Kuva 13 havainnollistaa tätä dilemmaa *etupään johtamisessa* (Front-end management). Luovuutta etupääninnovoinnissa on mahdollista edistää löysällä organistatorisella kurilla ja toimivalla henkilöstöhallinnalla. Tällaisessa tilanteessa työntekijöiden ei tarvitse kehittää uutta tuotetta markkinamenestys edellä, vaan keskittyä esimerkiksi tuotteen teknologisiin mahdollisuuksiin. Resurssien tehokas käyttö puolestaan saavutetaan vahvalla organisatorinen kurilla ja tiukoilla prosesseilla. Kuri ilmenee esimerkiksi siten, että tuotetta kehitetään markkinamenestys ja käyttäjäkokemus edellä.



Kuva 13: Etupääninnovoinnin dilemma [GSW06].

Optimaalinen ohjelmistokehitysprosessi mahdollistaa sekä luovuuden että tehokkuuden. Innovatiivisen ohjelmistotuotteen tehokas etupään kehitys vaatii elementtejä, jotka

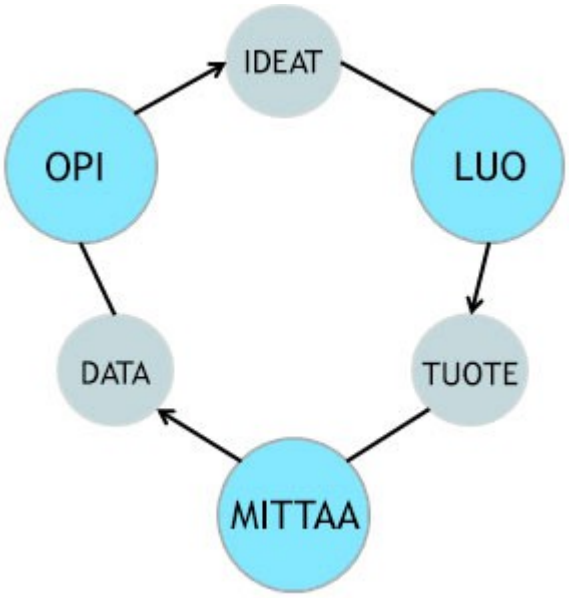
mahdollistavat sekä luovuuden että resurssien tehokkaan käytön [GSW06]. Lisäksi ohjelmistokehitysprosessin tulee tukea kulloista strategiaa vision saavuttamiseksi.

Tässä luvussa esitellään AdPearl Oy:lle räätälöity ohjelmistokehitysprosessi, joka perustuu Lean Startup -menetelmään sekä yhdistää luvussa 4 esitellyistä prosessimalleista sopivimmat ominaisuudet. Prosessi soveltuu ensimmäistä tuotettaan kehittäville yrityksille, joiden palvelun asiakaskunta ei ole vielä tiedossa sekä tuotteen *tilaaja* on yritys itse. Prosessissa käytetään Kanbania työnkulun hallintaan, XP:n toimintatapoja prosessin sisäisen kurin lisäämiseksi, Scrumin kehitysjonolistaa työtehtävien kirjaamiseksi sekä ketteristä menetelmistä tuttuja iteraatioita. Esiteltävässä prosessissa pyritään kiinnittämään erityishuomioita suunnitteluajatteluun, palvelumuotoiluun sekä käyttäjäinnovaatioiden aikaansaamiseen.

Ohjelmistokehitysprosessin pyrkimys on olla mahdollisimman ketterä ja välttää turhien ominaisuuksien toteuttamista Lean-ajattelun mukaisesti. Lähtökohta on, että kaikki mikä ei lisää arvoa asiakkaalle, on turha ottaa prosessissa huomioon. Niihin ominaisuuksiin, prosesseihin ja toimenpiteisiin – jotka eivät suoraan edistä AdPearl Oy:n ymmärrystä tuotteen toimivuudesta tai lisää asiakkaan saamaa arvoa – käytettävät resurssit pyritään minimoimaan.

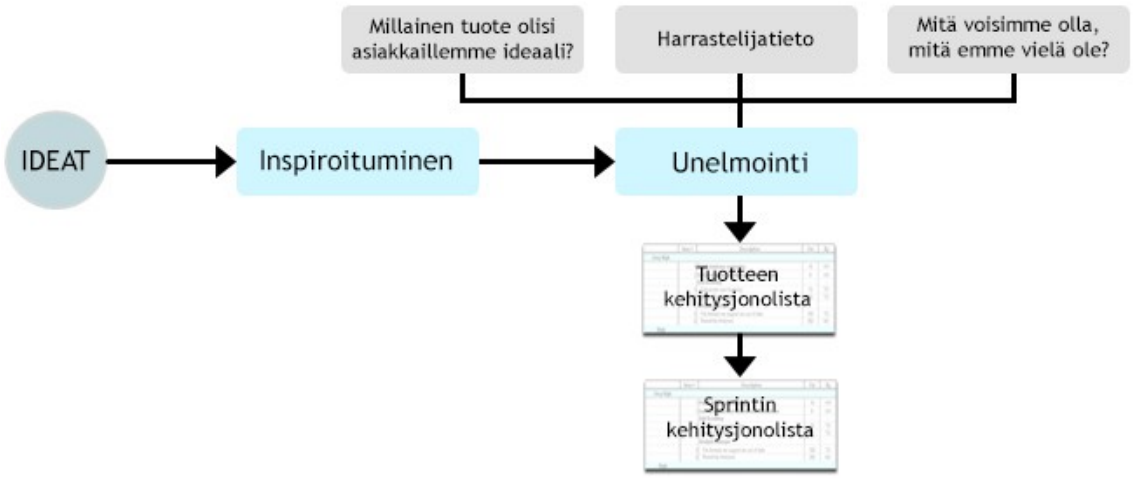
5.1 Prosessin ydin: luo-mittaa-opi

Kuvasta 14 nähdään, kuinka ohjelmistokehitysprosessin ydin pohjautuu Lean Startup -menetelmän luo-mittaa-opi -palautesilmukkaan. Silmukkaa pyritään läpikäymään niin nopeasti kuin mahdollista. Kun kaikki silmukan vaiheet on suoritettu, analysoidaan tuotteen toimivuudesta opittuja asioita ja päätetään kuinka tuotteen kehittämistä jatketaan. Prosessin ydin on seuraava: Ensiksi kartoitetaan ideat, jonka jälkeen luodaan minimaalinen toimiva tuote, jonka kelpoisuutta mitataan. Mittaa-vaiheessa katsotaan onko kohderyhmä saanut kehitetystä ominaisuudesta arvoa, kuten hypoteesissa oli määritelty. Tämä vaihe tuottaa dataa, jonka perusteella tuotteen toimivuudesta opitaan tulevia iteraatioita ajatellen. Opittuja asioita käytetään pohjana tulevaan iteraatioon: optimoidaanko tuotetta lisää vai koitetaanko uutta strategiaa vision saavuttamiseksi.



Kuva 14: Lean Startup -menetelmän palautesilmukka [Rie11].

Prosessi alkaa ideoiden tuottamisella. Jotta ideoinnista saa maksimaalisen hyödyn irti, pitää työntekijöiden olla inspiroituneita: pyrkimyksenä on ajatella suuria ja odottaa kehitettävältä tuotteelta menestystä. Kuvasta 15 huomataan, että ídeointiin käytetään Millerin innovaatioprosessia. Innovatiivisuutta kartoitetaan unelmoinnin avulla: ideoinnissa pyritään unelmoimaan suuria - mitä yritys voisi olla, mitä se ei vielä ole? Millainen tuote olisi asiakkaillemme ideaali? Mitä se tekisi ja kuinka se toimisi? Lisäksi jokaisen idean kohdalla mietitään tarkoin, onko ominaisuus välttämätön, toisin sanoen, tuoko se asiakkaalle lisäarvoa. Etenkin pienelle startup-yritykselle on luontevaa, että jokainen yrityksen työntekijä osallistuu ideointiin. Mahdollisia sisäisiä tietolähteitä, esimerkiksi sisäistä harrastelijatietoa, hyödynnetään mahdollisuuksien mukaan. Erityistä huomiota tulee kiinnittää myös asiakkaiden käyttökontekstin ja kulttuurin ymmärtämiseen: mitä asiakkaamme tarvitsevat ja kuinka asiakkaiden elämään helpotetaan yrityksen tuotteen avulla. Tuotetut ideat lisätään lopuksi tuotteen kehitysjonolistaan, josta ne päätyvät aikanaan sprintin kehitysjonolistaan.



Kuva 15: Luo-mittaa-opi -palautesilmukan IDEAT-vaihe.

5.1.1 Luo

AdPearl Oy:n prosessimallin perusta tulee Scrumista ja Kanbanista. Scrumin iteratiivinen ja inkrementaalinen tapa itsessään kannustaa olla ennustamatta tulevaisuutta liian pitkälle. Palautteen kerääminen asiakkailta kunkin Sprintin päätteeksi auttaa löytämään mahdolliset väärinymmärrykset nopeasti. Scrumtiimi keskittyy vain yhteen kehitysjonolistaan kerrallaan. Yhteen projektiin keskittyminen kerrallaan parantaa tuotteen laatua ja lisää tuotteliaisuutta. Scrum pyrkii lyhyillä sykleillään tuottamaan valmista ohjelmistoa mahdollisimman nopeasti.

Luo-vaihe aloitetaan suunnittelukokouksella, jonka tavoite on kerätä sprintin kehitysjonolistaan vaatimuksia tuotteen kehitysjonolistasta, joka on jo aiemmin syntynyt ideat-vaiheen tuloksena. Kehitysjonolistan sisältö kuvaa osaltaan yrityksen nykyisen strategian visionsa saavuttamiseksi. Listaan kirjataan järjestelmään toteutettavat ominaisuudet pieniin osatehtäviin ositettuina. Kullekin ominaisuudelle määritellään prioriteetti sekä kirjataan ominaisuuden arvioitu työmäärä. Sellaisille ominaisuuksille, joiden arvolupausta ei ole vielä varmistettu, määritellään *arvohypoteesi* (Value hypothesis). Arvohypoteesi on oletus siitä, kuinka paljon ominaisuus tuottaa asiakkaalle arvoa [Rie11]. Hypoteesi on hyvä kuvata kirjallisessa muodossa, määrittäen kohderyhmän, jolle tuotetta tai ominaisuutta kehitetään, sekä syyn, minkä takia kohderyhmän oletetaan tuotetta tai ominaisuutta käyttävän. Lisäksi

kullekin hypoteesille merkitään tavoite, kuinka monta prosenttia koko ominaisuuden kohderyhmästä ottaa ominaisuuden käyttöön. Hypoteesin määrittämiseen voi käyttää seuraavaa rakennetta:

Me uskomme, että [*luku*] prosenttia

[*ominaisuuden kohderyhmä*] käyttää [*ominaisuutta*]
[*syy/syyt*] ansiosta.

AdPearlin tapaukseen sopiva esimerkki hypoteesin määrittämisestä on seuraava: Me uskomme, että 15% rekisteröityneistä kauppiaista käyttää tuotealuekohtaista nostoa 100 euron lisämaksua vastaan sen antaman suuren CTR:n (Click Trough Rate) ansiosta.

Kehitysjonolistan perusteella toteutetaan ensin minimaalinen toimiva tuote. MVP:n ei ole tarkoitus olla täydellinen, vaan ainoastaan sellainen, että sen kattaa tuotteelle asetetut minimivaatimukset. Tällaisen, toisinaan hyvinkin vaatimattoman, tuotteen toteuttamisen ideana on testata tuotetta todellisilla asiakkailla ja kartoittaa tuotteen toimivuutta käytännössä. Nopea sykli ja varhaisessa vaiheessa toimivan tuotteen testaaminen oikeilla asiakkailla mahdollistaa tärkeiden oppimiskokemusten kartoittamisen tuotteesta. Näin vältetään tilanne, jossa pitkään kehitetyn tuotteen julkaisu paljastuu asiakkaiden toiveiden vastaiseksi. Tuotekehitys ei voi perustua ainoastaan siihen, mitä asiakkaat sanovat haluavansa. Tällainen vahvistettu oppiminen varmistaa, että asiakkaat ja käyttäjät saavat sitä mitä he todellisuudessa halusivat.

Kanban visualisoi kehityksen pullonkaulat. Suurin syy työtahdin hidastumiseen on usein se, että työntekijöillä on liian monta työtehtävää hoidossaan samanaikaisesti [Kni10]. Kanban suosii rajan asettamista samaan aikaan kehityksessä olevien työtehtävien määrälle. Rajojen asettaminen mahdollistaa sopeutumisen vallitseviin olosuhteisiin. Lisäksi Kanban-taulu mahdollistaa muutosten toteuttamiseen kesken kehityksen. Muutostarpeen havaitseminen ja muutosten mahdollistaminen on usein parempi vaihtoehto kuin se, että sprintin viedään loppuun vain todetakseen mitä saatiin aikaiseksi.

Kehitysjonolistan tehtävien toteuttamiseen käytetään Kanban-taulua, joka yhdistää tuotekehityksen tilat sekä Lean startup -menetelmän luo-mittaa-opi -prosessin tilat samaan tauluun [Zhe12]. Kanban-taulu visualisoi työnkulun ja rajoittaa työtehtävien liikakuormittamisen. Käynnissä olevat työtehtävät rajoitetaan yhtä työntekijää kohden kahteen. Optimitilanteessa kullakin työntekijällä on kullakin hetkellä vain yksi

työtehtävä; toinen tehtävä otetaan kehitykseen ainoastaan silloin, kuin ensimmäinen tehtävää ei sillä hetkellä voida suorittaa [Rie11]. Kuva 16 havainnollistaa esimerkin omaisesti AdPearl Oy:n työtehtävät, työtilat ja työnkulun Kanban-taulun avulla. Kirjaimet A-Ö kuvaavat työtehtäviä.

	Luo				**Mittaa / Opi**			
Kehitys-jonolista	Kehitys	Test.	Odott. julkaisua	**Tuotan-nossa**	Ei aiheuta haittaa	Paran-nuksen määrittä-minen	Odottavat julkaisua koko asiakaskun nalle	Vahv. Ominais.
W X Y Z Å Ä Ö	*S T U*	*Q R*	*N O P*	*I J K L M*	*G H*	*E F*	*C D E*	*A B*

Kuva 16: Yhdistetty luo-mittaa-opi Kanban-taulu [Zhe12].

Prosessiin sisältyy viisi XP:stä tuttua toimintatapaa kaoottisuuden estämiseksi:

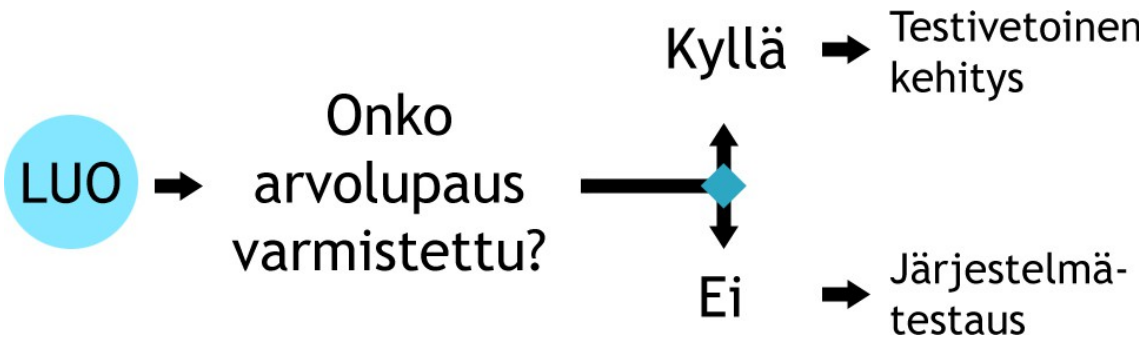
1. Jatkuva integraatio.
2. Refaktorointi.
3. Yhteiset koodauskonventiot.
4. Koodin yhteisomistajuus.
5. KISS-periaate.

Jatkuvalla integraatiolla tarkoitetaan sitä, että aina kun tehtävä on valmis, se integroidaan osaksi olemassa olevaa järjestelmää. Ohjelmakoodia refaktoroidaan jatkuvasti toiston poistamiseksi, kommunikoinnin parantamiseksi ja järjestelmän yksinkertaistamiseksi. Kukaan kehittäjä ei itse omista omaa koodiaan, vaan kaikki ohjelmakoodi on kaikkien kehittäjien yhteisomistajuudessa. Tällöin kuka tahansa voi muuttaa järjestelmän mitä koodia tahansa milloin tahansa. *KISS*-periaate (Keep It Short and Simple) on ajatusmalli, joka pyrkii siihen, että ohjelmisto rakennetaan niin

yksinkertaiseksi kuin mahdollista. Monimutkaisuutta poistetaan heti kun sellaista havaitaan.

Menetelmä sisältää viisi roolia: tuoteomistaja, tiimi, asiakas, käyttäjä ja sovellusasiantuntija. Roolit on johdettu Scrum- ja FDD-prosessimalleista. Tuoteomistaja vastaa projektin johtamisesta ja -hallinnasta. Tuoteomistaja on viimekädessä vastuussa projektista. Asiakaalla tarkoitetaan tahoa, joka tuotteen tilaaja tai maksaja. Tiimi on projektitiimi, joka osallistuu ideointiin ja kehitysjonolistalle tulevien ominaisuuksien luomiseen sekä niiden työmäärän arviointiin. Käyttäjällä tarkoitetaan sellaista henkilöä, joka on tuotteen loppukäyttäjä. Sovellusasiantuntija voi olla käyttäjä, asiakas, sponsori, liiketoiminta-analyytikko tai jokin näiden yhdistelmä. Sovellusasiantuntijan tehtävä on tarjota tietoa tuotteen sovellusalueen erityispiirteistä varmistaen, että projektitiimi kehittää oikeanlaisen tuotteen.

Testausmenetelmä valitaan sen perusteella, mitä ollaan toteuttamassa. Testausmenetelmän valinta on kuvattu kuvassa 17. *Testivetoinen kehitys* (Test Driven Development) on Extreme Programmingin toimintatapa, joka tarkoittaa yksikkötestin kirjoittamista samanaikaisesti, tai jopa ennen itse ohjelmakoodin kirjoittamista [Bec00]. Testivetoinen kehitys tuottaa koodia, jonka testikattavuus on todella korkeaa. Testivetoista kehitystä käytetään AdPearl Oy:lle räätälöidyssä prosessissa silloin, kun toteutetaan ominaisuutta, jonka arvolupaus on varmistettu. Käytännössä tämä tarkoittaa lähinnä uusien sisäisten ominaisuuksien toteuttamista, kuten esimerkiksi jo olemassa olevan minimalistisen toimivan tuotteen jonkin ominaisuuden parantamista. MVP-vaiheessa testivetoista kehitystä ei käytetä siitä syystä, että testivetoinen kehitys hidastaa merkittävästi palautesilmukan läpikäyntiä. Kattavien yksikkötestien kirjoittaminen vie runsaasti resursseja, joten ainoastaan liiketoiminnan kannalta olennaisimmille osille toteutetaan yksikkötestit. Muutoin ominaisuuden julkaisukelpoisuus varmistetaan järjestelmätestauksella, eli yksikkötestaamisen sijaan testataan laajemman kokonaisuuden julkaisukelpoisuutta. Testauksen voi tehdä myös tarvittaessa integrointitestauksena, jolloin jokin osajärjestelmä testataan varmistuakseen siitä, että osat yhdessä muodostavat toimivan kokonaisuuden.



Kuva 17: Testausmenetelmän valinta.

Prosessi sisältää viisi dokumentointimuotoa:

1. Tuotteen kehitysjonolista sisältää kaikki ominaisuudet, joista otetaan kunkin iteraation alussa prioriteetiltaan korkeimmat tehtävät sprintin kehitysjonolistaan. Tehtävät voivat olla joko suunnittelutehtäviä, ohjelmointitehtäviä, testaustehtäviä, dokumentointitehtäviä, vaatimuksia, tarinoita tai ominaisuuksia.
2. Sprintin kehitysjonolista pitää sisällään tuotteen kehitysjonolistasta poimitut tehtävät, jotka kyseisen iteraation aikana halutaan toteuttaa.
3. Yhdistetty Kanban-taulu, joka pitää sisällään pelkistetyn sprintin kehitysjonolistan sekä visualisoi työnkulun.
4. Tuotettu ohjelmakoodi.
5. Muu projektin aikana tuotettu suunnitteludokumentaatio.

5.1.2 Mittaa

Kun uuden ominaisuudet on viety tuotantoon, yrityksen on mitattava ominaisuuksien arvolupaukset. Tässä vaiheessa katsotaan, onko kohderyhmä saanut kehitetystä ominaisuudesta arvoa, kuten hypoteesissa oli määritelty.

Ominaisuuksien kehittyminen voidaan visualisoida taulukon avulla [Rie11]. Taulukkoon 7 on merkitty esimerkinomaisesti AdPearl Oy:n neljän merkittävän hypoteesin prosentuaalinen toteutumisaste suhteessa ominaisuuden kohderyhmään. Tavoite-sarakkeeseen on kirjattu käyttöasteen tavoite prosentteina, joka optimointikierroksen jälkeen halutaan saavuttaa. Keskimmäiseen sarakkeeseen on merkitty kunkin hypoteesin käyttöaste minimaalisen toimivan tuotteen jälkeen. Oikeanpuoliseen sarakkeeseen on merkitty prosenttiosuus kaikista käyttäjistä ensimmäisen optimointikierroksen jälkeen.

Taulukon prosentuaalinen kehittyminen antaa suuntaa yritykselle, kuinka loppukäyttäjät omaksuvat palvelun ominaisuudet.

	Tavoite	**MVP**	**Optimoinnin jälkeen**
Liittyminen kauppiaaksi	10,00%	5,00%	20,00%
Tunnuksen aktivointi	80,00%	50,00%	80,00%
Aktivoitunut kauppias lisää tarjouksen	95,00%	60,00%	90,00%
Aktivoi maksullisen palvelun	5,00%	Liian alhainen	5,00%

Taulukko 7: Kehitettyjen ominaisuuksien prosentuaaliset toteutumiset.

Taulukko helpottaa tavoitteiden seuraamista. Mikäli tavoitteen toteutuminen ei edisty optimoinneista huolimatta halutulla tavalla, voi ainoa vaihtoehto olla täyskäännöksen tekeminen. Mittaamiseen voidaan käyttää myös *joukkoanalyysia* (Cohort analysis), jossa vertaillaan tietyn ajanjakson uusien asiakkaiden käyttäytymistä aikaisempiin asiakkaisiin tai *joukkoanalyysiä* (Split analysis), jossa uutta tuotevarianttia testataan osalla asiakkaista, verraten heidän käyttäytymistään suhteessa aiempaa varianttia käyttäviin asiakkaisiin [Rie11].

5.1.3 Opi

Opi-vaiheessa analysoidaan uuden tuotteen tai ominaisuuden toimivuudesta mitattuja asioita ja päätetään kuinka tuotteen kehittämistä jatketaan. Mahdollisuuksia on kaksi: joko optimoida tuotetta lisää tai tehdä muutoksia. Muutoksen tekeminen vaatii liiketoimintamallistrategian muutoksia yrityksen vision saavuttamiseksi. Kun strategia muuttuu, usein myös kehitettyyn ohjelmistoon tehdään merkittäviä muutoksia tai se jopa hylätään kokonaan uuden ohjelmiston tieltä. Täyskäännöksiä on useita: tarkennusmuutos, loitonnusmuutos, asiakassegmenttimuutos, asiakastarvemuutos, alustan muutos, liiketoiminta-arkkitehtuurin muutos, kasvustrategian muutos, kanavamuutos tai teknologiamuutos. Täyskäännökset on esitelty yksityiskohtaisemmin luvussa 2.

Muutoksen tarve havaitaan optimointitulosten perusteella. Jos optimoinnit eivät ole tuottaneet haluttua tulosta, on muutos yrityksen ainoa oikea tie. Vastaukset strategian sisältämien oletusten löytyvät mittaamisen tuloksena, sekä kommunikoimalla

asiakkaiden kanssa.

5.2 Lean Startup -ohjelmistokehitysprosessin soveltuvuus

Esitelty ohjelmistokehitysprosessi on räätälöity AdPearl Oy:n tarpeisiin, ottaen huomioon yrityksen vaatimukset ohjelmistokehitysprosessille. Prosessi on suunniteltu etenkin liiketoimintamallin vaatimukset silmällä pitäen.

Liiketoimintamallivaatimuksista luo-mittaa-opi -prosessi täyttyy, sillä koko ohjelmistokehitysprosessi on rakennettu sen ympärille. Prosessi on myös ketterä, sillä luo-mittaa-opi -prosessia pyritään läpikäymään niin nopeasti kuin mahdollista. Varsinaista aikamäärettä iteraatioille ei ole. Lisäksi prosessi on luotu reagoimaan nopeasti muutoksiin, sekä olemaan kykenevä muuttamaan strategiaa nopealla aikavälillä.

Suunnitteluajattelun, palvelumuotoilun ja avoimen innovoinnin yhdistäminen ohjelmistokehitysprosessiin on jokseenkin hankalaa. Menetelmät pohjautuvat kokonaisvaltaisen tuotteen rakentamiseen. Näitä menetelmiä voi käyttää hyväksi kehitysprosessin ideointi-vaiheessa. Suunnitteluajattelun idea on ymmärtää tuoteominaisuuksien sijaan tuotteen tai palvelun käyttökonteksti ja -kulttuuri. Menetelmä pyrkii prototyyppien avulla paljastamaan ideoiden vahvuudet ja heikkoudet. Prototyyppien tuottaminen on mahdollista yhdistää ohjelmistokehitysprosessiin. Ketterät menetelmät eivät tähän varsinaisesti kannusta, sillä näissä menetelmissä iteraation jälkeinen julkaisu nähdään *valmiina* tuotteena tai tuotteen osana, ei keinona vahvistaa asetettuja oletuksia käyttökontekstista tai kulttuurista. Räätälöity ohjelmistokehitysprosessi sen sijaan pyrkii tarttumaan juuri tähän luo-mittaa-opi -prosessillaan. Suunnitteluajattelu pyrkii hyödyntämään kuluttajien tietämystä tuotekehityksessä, jolloin on hankalaa kuvitella valmista tuotetta, ilman että sitä on testattu oikeilla asiakkailla. Tuotteen tilaaja, asiakas, on useissa ketterissä menetelmissä läsnä kehitystyössä, mutta asiakas ei välttämättä ole tuotteen loppukäyttäjä.

Palvelumuotoilu on osaltaan samankaltainen kuin suunnitteluajattelu – menetelmässä keskitytään suunnittelemaan tuote palvelunkaltaisesti, asiakkaan perspektiivistä, kaikki sidosryhmät huomioiden. Erona suunnitteluajatteluun on tapa nähdä tuote kokonaisuutena – prosessi voi alkaa ennen varsinaista tuotteen käyttöä ja prosessi jatkuu tuotteen varsinaisen käytön jälkeen. Palvelumuotoilu on hankala liittää osaksi ohjelmistokehitysprosessia, sillä sen menetelmät eivät ole kovin konkreettisia.

Ainoastaan kaikkien sidosryhmien väliset suhteet voidaan kuvata sidosryhmäkartoilla, jonka avulla on tunnistettavissa kuinka paljon arvoa kukin osapuoli tuottaa kullekin osapuolelle.

Myös avoimen innovoinnin yhdistäminen ohjelmistokehitysprosessin menetelmiin on hankalaa. Avoin innovointi on enemmänkin toimintatapa, jossa yritys lähtökohtaisesti joko panostaa tai on panostamatta esimerkiksi avoimen lähdekoodin projekteihin tai innovaatioalustana toimimiseen. Ohjelmistokehitysprosessiin on vaikea sisällyttää menetelmiä, jotka kannustavat avoimeen innovointiin.

Harrastelijatieto on helppo yhdistää kehitysprosessiin samoin kuin Millerin innovaatioprosessi. Molemmat menetelmät on yhdistetty prosessin ideat-vaiheeseen. Suunnitteluvaihe sisältyy myös kaikkiin muihin tutkielmassa mainittuihin ketterät menetelmiin. Harrastelijatieto tosin vaatii, että yrityksestä löytyy henkilöitä, joilla on tietoa kehitettävän tuotteen aihealueesta. Millerin innovaatioprosessin unelma-vaihe yhdistää suunnitteluajattelun ja palvelumuotoilun ajattelu perspektiiviä ja sopii hyvin suunnittelukokouksen innovointimenetelmäksi. Innovaatioprosessin suunnitelu-vaihe sopii yhteen ohjelmistokehitysprosessien suunnittelu-vaihdeiden kanssa.

AdPearl Oy:lle räätälöity ohjelmistokehitysprosessi	
Vaatimus	**Täyttyykö vaatimus (Kyllä / Osittain / Ei)**
Liiketoimintamalli	
Luo-mittaa-opi -prosessi	**Kyllä**
Ketteryys	**Kyllä**
Turhuuden karsiminen	**Kyllä**
Skaalautuvuus	**Osittain**
Innovointi	
Unelmointi, ideointi	**Kyllä**
Prototyyppien jatkuva tuottaminen	**Kyllä**
Käyttäjäkokemuksen huomioiminen	**Osittain**
Palvelun tuotteistaminen	**Osittain**
Täydellinen palvelukokemus	**Kyllä**
Alusta uusille, kolmannen osapuolen innovaatioille	**Ei**
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	**Kyllä**

Taulukko 8: AdPearl Oy:n vaatimusten toteutuminen räätälöidyn prosessimallin avulla.

6 Analyysi

Tässä luvussa vastataan tutkielman tutkimuskysymyksiin sekä mainitaan tutkielman rajoitteista ja tarvittavasta jatkotutkimuksesta.

6.1 Vastaukset tutkimuskysymyksiin

Tutkielman ensimmäinen tutkimuskysymys on ”kuinka ketterät menetelmät sopivat yhteen startup-yritysten innovaatioprosssien kanssa?” Ketterät menetelmät sopivat periaatteeltaan hyvin yhteen startup-yritysten innovaatioprosessien kanssa. Ketterät menetelmät ovat iteratiivisia ja kykenevät reagoimaan nopeasti muutoksiin. Ohjelmistoyrityksen liiketoimintamalli on startup-yrityksen näkökulmasta ainoastaan joukko arvauksia, jotka pitää mahdollisimman nopeasti vahvistaa oikeiksi tai vääriksi. Lean Startup -menetelmä on tähän ongelmaan toimiva ratkaisu, joka on yhdistettävissä olemassa oleviin ohjelmistokehitysprosesseihin. Lean Startup -menetelmä pakottaa yritystä vahvistamaan omista tuotteistaan tai uusista ominaisuuksista luodut oletukset. Kun tuote tai uusi ominaisuus tuottaa vahvistetun oppimisen tuloksia, voidaan silloin puhua todellisista innovaatioista. Ketterissä menetelmissä ohjelmiston toimivuus on tärkeämpää kuin tuotettu dokumentaatio, jolloin uusien liiketoimintamallien toimivuus voidaan nopeasti joko vahvistaa tai kumota.

Tutkielmassa esitellyistä innovaatiomenetelmistä parhaiten ohjelmistokehitysprosessiin sopivat Millerin innovaatioprosessi, palvelumuotoilu, harrastelijatiedon hyödyntäminen sekä luo-mittaa-opi -prosessi. Palvelumuotoilua tai avointa innovointia on vaikea yhdistää osaksi kehitysprosessia.

Tutkielman toinen tutkimuskysymys on ”mikä ketterä menetelmä sopii ominaisuuksiltaan parhaiten?” ja kolmas ”millainen ketterä menetelmä tukee innovaatioprosesseja parhaiten?” Kaikki menetelmät FDD:tä lukuunottamatta soveltuvat AdPearl Oy:lle tärkeimmän innovaatioprosessin, Lean Startup -menetelmän luo-mittaa-opi -prosessin, suorittamiseen. Taulukkoon 9 on koottu yhteen tutkielmassa esiteltyjen prosessimallien vaatimusten toteutuminen. AdPearl Oy:n vaatimusten kannalta paras ketterä menetelmä on Scrum, joka on joustava ja sopii hyvin yhteen Lean Startup -menetelmän luo-mittaa-opi -prosessin kanssa. AdPearl Oy:lle räätälöity ohjelmistokehitysprosessi pohjautuu Scrumiin, lisäten siihen elementtejä Kanbanista ja

Extreme Programming-menetelmästä sekä yhdistäen siihen innovaatiomenetelmiä. Tutkielman huonoin prosessimalli AdPearl Oy:n tarpeisiin on Feature Driven Development. FDD vaatii tarkkaa etukäteistä tietoa tuotteen toimintaympäristöstä, eikä se siksi sovellu startup-yrityksen prosessimalliksi.

Kaikki menetelmät noudattavat ainakin osittain Lean ajattelun periaatteita. XP sisältää tiukat ohjailevat toimintatavat, jotka tekevät prosessista omalta osaltaan raskaan. Kaikki XP:n toimintatavat eivät tuota asiakkaalle arvoa, mikä sotii Lean ajattelun periaatteita vastaan. XP on Scrumia ohjailevampi prosessimalli. Se pitää sisällään suuren osan Scrumin periaatteista, sekä lisäksi tarkkoja toimintatapoja, kuten tiukat testauskäytännöt ja pariohjelmoinnin. Scrum taas on puolestaan ohjailevampi kuin Kanban, sisältäen selkeän määrätyn iteraatiorakenteen. Kanban sisältää vain prosessin työnkulun hallintaan jättäen lähes kaiken muun avoimeksi, vaatien vain työnkulun visualisoinnin ja työmäärän rajoittamisen.

Yrityksen, kehitettävän tuotteen ja kehitystiimin kasvuvaatimuksiin vastaa parhaiten Scrum, FDD sekä AdPearl Oy:lle räätälöity ohjelmistokehitysprosessi. XP vaatii kehitystiimin tiukkaa kommunikointia eikä sisällä menetelmiä tiimien yhteistyöhön. Kanban puolestaan visualisoi työnkulun yleensä yhden Kanban-taulun avulla. Kanban ei itsessään ole kovinkaan kattava prosessimalli, joten sen vertaaminen tässä suhteessa on rajoittunutta. Kanban toimii parhaiten työkaluna jonkin toisen prosessimallin sisällä.

Yhteenveto ohjelmistokehitysprosesseista					
Vaatimus	**Täyttyykö vaatimus (Kyllä / Osittain / Ei)**				
	XP	Scrum	Kanban	FDD	Räätälöity
Liiketoimintamalli					
Luo-mittaa-opi -prosessi	**Kyllä**	**Kyllä**	**Kyllä**	**Osittain**	**Kyllä**
Ketteryys	**Kyllä**	**Kyllä**	**Osittain**	**Osittain**	**Kyllä**
Turhuuden karsiminen	**Osittain**	**Kyllä**	**Kyllä**	**Kyllä**	**Kyllä**
Skaalautuvuus	**Osittain**	**Kyllä**	**Ei**	**Kyllä**	**Osittain**
Innovointi					
Unelmointi, ideointi	**Kyllä**	**Kyllä**	**Osittain**	**Kyllä**	**Kyllä**
Prototyyppien jatkuva tuottaminen	**Kyllä**	**Kyllä**	**Kyllä**	**Osittain**	**Kyllä**
Käyttäjäkokemuksen huomioiminen	**Osittain**	**Osittain**	**Ei**	**Kyllä**	**Osittain**
Palvelun tuotteistaminen	**Ei**	**Osittain**	**Ei**	**Ei**	**Osittain**
Täydellinen palvelukokemus	**Ei**	**Osittain**	**Ei**	**Kyllä**	**Kyllä**
Alusta uusille, kolmannen osapuolen innovaatioille	**Ei**	**Ei**	**Ei**	**Ei**	**Ei**
Yrityksen sisällä olevan, epäsuoran tietämyksen tunnistaminen ja hyödyntäminen	**Kyllä**	**Kyllä**	**Osittain**	**Kyllä**	**Kyllä**

Taulukko 9: AdPearl Oy:n vaatimusten [taulukko 1] toteutuminen tutkielmassa esiteltyjen prosessimallien avulla.

6.2 Rajoitukset ja tulevaisuuden työ

Tutkielma on luonteeltaan kirjallisuuskatsaus, joka vaatii tuekseen käytännön tutkimustuloksia. Luvussa 5 esiteltyä AdPearl Oy:lle räätälöityä prosessia ei tämän tutkielman osalta ehditty tutkia käytännön tilanteissa. Prosessin soveltuvuuden kannalta olisi tärkeää saada tutkimustietoa prosessin toimivuudesta myös käytännön ohjelmistoprojekteissa. Tutkielmassa muodostettua analyysia sopivimmasta ohjelmistokehitysprosessista innovatiiviselle startup-yritykselle voidaan kuitenkin käyttää menetelmävalintaohjeena uusille – ensimmäistä tuotettaan kehittäville – ohjelmistoyrityksille.

7 Yhteenveto

Tutkielmassa tutkittiin ketterien ohjelmistomenetelmien toimivuutta startup-yrityksen innovaatioprosessissa. Ohjelmistoalalle sopivia innovaatiomenetelmiä ovat suunnitteluajattelu, palvelumuotoilu, avoin innovointi, käyttäjäinnovaatiot, Millerin innovaatioprosessi sekä Lean Startup -menetelmä.

Ensimmäistä tuotettaan kehittävien startup-yritysten yhdistävä tekijä on epävarmuus. Perinteiset vesiputousmalliin pohjautuvat menetelmät eivät ole parhaimmillaan vastaamaan äkillisiin muutoksiin. Tästä johtuen startup-yrityksille paremmin soveltuvia ohjelmistomenetelmiä ovat nopeaan reagointiin kykenevät ketterät menetelmät. Yleisiä ketteriä menetelmiä ovat Extreme Programming, Scrum, Kanban ja Feature Driven Development. Ensimmäistä tuotettaan kehittävän startup-yrityksen, AdPearl Oy:n, vaatimusten kannalta paras ketterä menetelmä on Scrum, joka on joustava ja sopii hyvin yhteen Lean Startup -menetelmän luo-mittaa-opi -prosessin kanssa. Edellä mainittujen yleisesti tunnettujen ketterien menetelmien lisäksi tutkielmassa esitelty Lean Startup -menetelmään pohjautuva – AdPearl Oy:n tarpeisiin sovelettu – prosessimalli, yhdistää olemassa olevien prosessimallien parhaat piirteet vastaten parhaiten yrityksen asettamiin liiketoimintamalli- ja innovaatiovaatimuksiin.

Lähteet

ApT06 Apilo, T. ja Taskinen, T., 2006, Innovaatioiden johtaminen. *VTT Tiedotteita # Research Notes 2330*, Espoo.

ASR02 Abrahamsson, P., Salo, O., Ronkainen, J. ja Warsta, J., 2002. Agile software development methods – Review and analysis, *VTT Publications 478*. [Myös http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf].

BBB01 Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. ja Thomas, D., 2001, Manifesto for Agile Software Development, *http://agilemanifesto.org* [1.5.2012].

Bec00 Beck, Kent, 2000, Extreme Programming Explained – Embrace Change, *Addison-Wesley*, New York.

BoL09 Boudreau K. J. ja Lakhani K. R, 2009, How to Manage Outside Innovation: Competitive Markets or Collaborative Communities? *Sloan Manage. Rev.* Vol. 50, No. 4, sivut 69-76.

Bro08 Brown, Tim, 2008, Design Thinking, *Harvard Business Review*, Vol. 86, No. 6, sivut 84-93.

Che05 Chesbrough, Henry, 2005, Open Innovation: A New Paradigm for Understanding Industrial Innovation, *Oxford University Press*.

GSW06 Gassmann, O., Sandmeier, P., Wecht, C, H., 2006, Extreme customer innovation in the front-end: learning from a new software paradigm, *Int. J. Technology Management*, Vol. 33, No. 1., sivut 46-65.

Hir08 Hiranabe, K., 2008, Kanban applied to software development: From agile to lean, http://www.infoq.com/articles/hiranabe-lean-agile-kanban.

IEE04 IEEE Computer Society, 2004, Guide to the Software Engineering Body of Knowledge, *SWEBOK*.

IKN10 Ikonen, M., Kettunen, P., Nilay, O. ja Abrahamsson, P., 2010, Exploring the Sources of Waste in Kanban Software Development Projects, *Proc. of SEAA Euromicro (IEEE)*.

JyR12 Veerapaneni Esther Jyothi et al., 2012, Effective Implementation of Agile Practices – Incoordination with Lean Kanban, *International Journal on Computer Science and Engineering (IJCSE)*, Vol. 4 No. 01, sivut 87-91.

Kni10 Kniberg, Henrik, 2010, Kanban vs Scrum – How to make the most of both, *http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf* [20.5.2012].

Kot07 Kotro, Tanja, 2007, User Orientation Through Experience: A Study of Hobbyist Knowing in Product. *An Interdisciplinary Journal on Humans in ICT Environments*, Vol 3, ISSN: 1795-6889, sivut 154-166.

Lik04 Liker, Jeffrey K. , 2004, The Toyota Way, *McGraw-Hill, USA*.

MAL08 Mäkitalo-Keinonen, T., Arenius, P., Liikala, S., 2008, Käyttäjät ja yritykset innovaatioyhteistyössä. *Julkaisusarja A –Turun kauppakorkeakoulu, Porin yksikkö*. Nro 26/2008.

Mil01 Miller, Granville G., 2001, The Characteristics of Agile Software Processes, *The 39th International Conference of Object-Oriented Languages and Systems (TOOLS 39)*, Santa Barbara, CA.

Mil09 Miller, Lawrence M., 2009, Creating the Best Company Ever, A Model for Deploying Internal Strategy. *EBBF Annual Conference*.

Mol06 Molin-Juustila, Tonja, 2006, Cross-functional interaction during the early phases of user-centered software new product development: reconsidering the common area of interest, *Acta Univ. Ouluensis A 455. Oulu: Oulu University Press,* ISSN: 1796-220X.

OsP10 Osterwalder, A., Pigneur, Y., 2010, Business Model Generation, *John Wiley & Sons, Inc*.

PaF02 Palmer S., Felsing J., 2002, Practical Guide to Feature-Driven Development. *Upper Saddle River*, New Jersey: Prentice Hall, Inc.

Pin09 Pink, Daniel, 2009, The Surprising Truth About What Motivates Us, *New York: Riverhead Books*

Reh11 Rehn, Alf, 2011, Dangerous Ideas - When provocative thinking becomes your most valuable asset, *Marshall Cavendish International*.

Rie11 Ries, Eric, 2011, The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. *New York: Crown Business*.

Rog62 Rogers, Everett M., 1962, Diffusion of Innovations, *Free Press, New York,* 5. edition.

Rog83 Rogers, Everett M., 1983, Diffusion of Innovations, *Free Press, New York*.

Roy70 Royce, W. W., 1970, Managing the developmentof large software systems: Concepts and techniques. *In Proceedings of IEEE WESTCON*. Los Angeles,

CA, 1–9.

SBT09 Shalloway, A., Beaver, G. ja Trott, J. R., 2009, Lean-Agile Software Development: Achieving Enterprise Agility, *Addison-Wesley*, Boston.

ScB02 Schwaber, K. ja Beedle, M., 2002, Agile Software Development With Scrum, *Upper Saddle River*, Prentice-Hall, NJ.

Sch39 Schumpeter, Joseph, 1939. Business Cycles. A Theoretical, Historical, and Statistical Analysis of the Capitalist Process. Volume I. *McGraw-Hill Book Company, Inc*.

SDN12a Service Design Network, 2012, http://www.service-design-network.org/content/sdn-manifesto, [7.2.2012].

SDN12b Servide Design Network, 2012, Service Design as an emerging field, http://www.service-design-network.org/system/files/media/Final-Service%20Design%20as%20an%20emerging%20field.pdf, [7.2.2012].

SDW04 Santos, J., Doz, Y. ja Williamson, P., 2004, Is your Innovation Process Global? *Mit Sloan Management Review*, Vol 45, No.4, sivut 31-37.

TaN86 Takeuchi, H. ja Nonaka, I, 1986, The New New Product Development Game, *Harvard Business Review,* Jan./Feb., sivut 137-146.

WJR90 Womack, J., Jones, D. ja Roos, D., 1990, The Machine that Changed the world: The Story of Lean Production, *Harper Collins,* New York.

Zhe12 Zheglov, Alexei, 2012, Kanban and Lean Startup: Making the Most of Both, http://www.agilejournal.com/articles/columns/column-articles/6511-kanban-and-lean-startup-making-the-most-of-both, [2.7.2012].